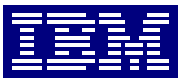
USCENSUSBUREAU
Helping You Make Informed Decisions



ADVANCED QUERY TECHNICAL DESIGN

DADS Project

Version No: 3.8

Version Publishing Date: December 7, 2006

Contract Number: 50-YABC-7-66012

TABLE OF CONTENTS

1. PREFACE

1.1. Scope

This document is a comprehensive guide to the Advanced Query (AQ) application, one of the major systems in the Data Access & Dissemination Systems (DADS) program.

1.2. Audience

The target audiences for this document are the developers who maintain and enhance AQ.

1.3. Purpose

This document describes the technical design of AQ and contains the information necessary for technical resources working on AQ to understand the design that serves as the foundation of AQ.

2. INTRODUCTION

In 1998 the U.S. Census Bureau launched a study into feasibility of automating the production of tabulations from the full Census 2000 microdata files with confidentiality protection. At the time, the Census Bureau relied on human intervention to review special tabulations derived from the Decennial Census microdata files before releasing those tabulations to the public. The increased use of the Internet to access Census data spurred the search for automated solutions that could create tabulations on-the-fly while enforcing disclosure limitations normally imposed by statisticians. During the study, an Oracle test system was built to experiment with programming disclosure rules into SQL (Sequence Query Language for relational data bases). After validating that disclosure could be automated, the Census Bureau's Disclosure Review Board approved the confidentiality business rules developed during the study, stating that these rules were sufficient to protect confidentiality.

The purpose of the Advanced Query (AQ) system is to allow authorized Census Bureau employees and government agency employees to query HDF and SEDF microdata via the Internet and Census Intranet. Prior to launching AQ, microdata recipients were responsible for developing their own software programs to tabulate and run statistical reports on the microdata. The AQ system simplifies the process for many users by providing both the microdata and a tool to analyze it.

AQ is a web-based application powered by MicroStrategy's OLAP engine and a DB2 data warehouse that houses HDF and SEDF microdata. Since the AQ system is web-based, it provides a cost-effective means of distribution to the Census Bureau's user base.

The documentation that follows describes the technical design for the Advanced Query application and its components. This *Technical Design* document is intended to provide application developers with sufficient background information to update and maintain the Advanced Query application and assumes familiarity with MicroStrategy platform and knowledge of the following programming languages:

- Microsoft Active Server Pages (ASP)
- Microsoft Visual Basic 6.0 (VB)
- JavaScript
- Java
- XML (eXensible Markup Language)

In addition, this document assumes the reader has a general understanding of databases and has read the High Level System Architecture document.

3. SYSTEM DESIGN

3.1. Overview

As described in the High Level System Architecture documentation, the AQ system is designed to produce on-line cross-tabulations of demographic characteristics for HDF and SEDF microdata. The sections that follow describe the logical data structure in greater depth than the High Level System Architecture and with emphasis on how MicroStrategy is configured to deliver the data. Some information in the High Level System Architecture is duplicated here for clarity.

3.1.1. MicroStrategy Project Structure: HDF vs. SEDF

The High Level System Architecture guide identifies HDF and SEDF microdata as the logical data structures in AQ. The MicroStrategy Intelligence Server is configured to implement these logical data structures as two separate MicroStrategy projects: an HDF project and an SEDF project. The HDF and SEDF MicroStrategy projects do not share data. Each project has its own set of MircoStrategy objects, including: attributes, filters, metrics, prompts, functions and facts; all sourced from separate HDF and SEDF database instances.

There were several motivations for creating separate HDF and SEDF MicroStrategy projects. First, the Census Bureau delivers HDF and SEDF survey samples separately, making it easier to maintain them as separate projects. Second, the HDF and SEDF record sets contain different information, making it necessary to create some database tables unique to each project. Finally, database performance was significantly improved by segregating HDF and SEDF data. These issues are discussed in more detail in the database documentation for AQ.

- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the AQ database documentation.*

3.1.2. User Groups

The Advanced Query application was designed to disseminate microdata analysis to two user groups:

- External users
- Internal users

External users can only view microdata cross-tabulations that have passed through strict **confidentiality filters** to remove records that might enable the user to identify information about a specific individual. Census Bureau employees and employees of government agencies such as State Data Centers and Census Information Centers, are examples of external users.

Internal users can view microdata cross-tabulations with relaxed restrictions. This set of users is limited to a small group of Census Bureau employees who have sworn to uphold Title 13 confidentiality and need access to unfiltered data as part of their work.

External vs. internal use of the AQ system is enforced by password protecting the external and internal AQ web sites. Passwords are created on the Intelligence Servers for the external and internal AQ web sites and are stored in separate MicroStrategy metadata database instances, thus preventing cross-over access. In addition, the external and internal AQ systems are segregated on the network so that they not only reside on different web and Intelligence servers but also are on different network segments. Finally, external vs. internal use of AQ is enforced by application coding techniques that utilize software filters to hide confidential data from external users.

Please see sections 4.2.3 and 5 of this document for a more detailed discussion of these techniques.

3.2. AQ System Design

The AQ system is made up of four components:

1. Web Server (IIS 5.0)
2. MicroStrategy Intelligence Server (v 7.2.2)

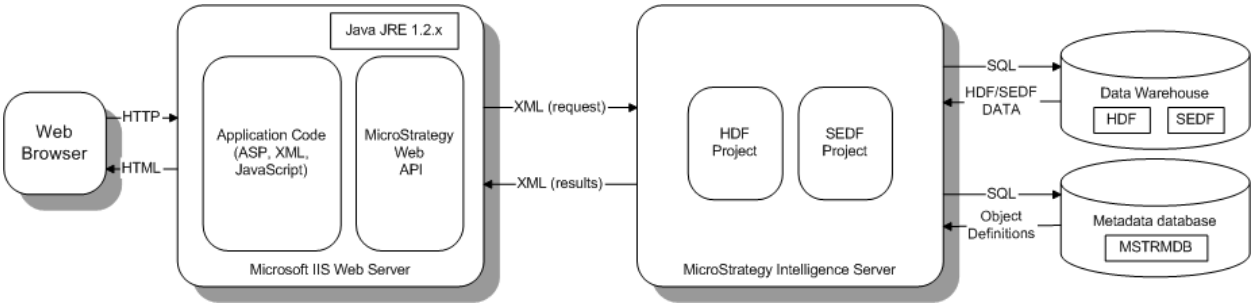
3. Data Warehouse Server
4. MicroStrategy Metadata Server

Together these components enable users to select criteria for HDF or SEDF cross-tabulations and view results. Operationally, these components have different roles within the AQ system, communicating with each other to fulfill a user's request.

3.2.1. Communication Flow Between Components

Figure 1 below depicts the communication flow between AQ System components.

Figure 1: Communication Flow Between Components



When a user connects to the AQ web site using a browser, the user builds a data request by navigating through ASP pages on the web server to select criteria for cross-tabulation. After a user submits criteria, the web server redirects the request for HDF or SEDF data to the MicroStrategy Intelligence server.

The MicroStrategy Intelligence server processes user requests and generates SQL queries that are executed against data in the data warehouse. The MicroStrategy Intelligence server also generates and executes SQL queries against its own metadata database where Intelligence server configuration information is stored.

Finally, the MicroStrategy Intelligence server returns the data to the web server; and the web server, sends the data to the client browser where it is displayed to the user.

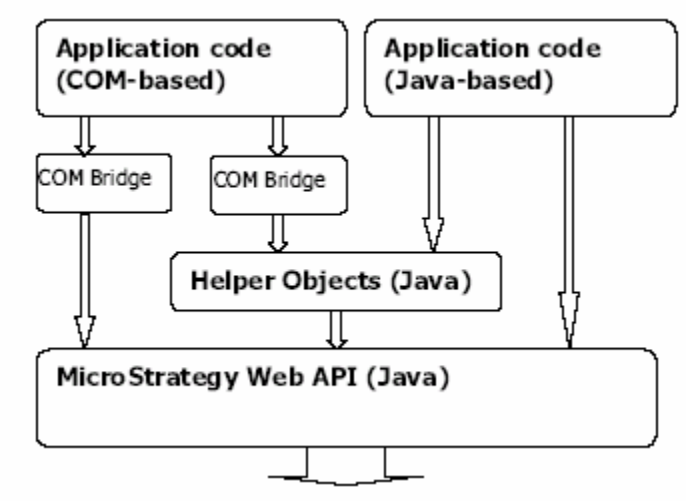
The sections that follow describe each component's internal processing in detail.

3.2.2. Web Server Component

All web servers are designed to handle HTTP requests. However, in a MicroStrategy environment, a web server must be able to translate HTTP requests into a proprietary XML format understood by the MicroStrategy Intelligence Server. This ability is added to AQ's IIS 5.0 web server by installing MircoStrategy's Web API through which standard requests are processed. The API is not part of the standard MicroStrategy installation and must be downloaded from MicroStrategy's web site and installed on the web server.

The web server forwards client requests to the MicroStrategy Intelligence server using Active Server Pages to call API methods provided by MicroStrategy. The MicroStrategy API is a set of Java classes requiring a Java run-time environment of 1.2.2 or higher installed on the web server. Since ASP pages are based on VBScript and cannot directly access Java, MicroStrategy provides a set of COM DLLs, which utilize the Java Native Interface (JNI) standard to form a bridge between the COM environment and the Java API. The API converts requests into XML before transmitting them to the Intelligence server via a socket connection created by the API. **Figure 2** below depicts the structure of the MicroStrategy Web API.

Figure 2: MicroStrategy Web API



Source: MicroStrategy

For Additional Details on this topic:

- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the MicroStrategy Web Universal Installation and Deployment Guide.*

- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the MicroStrategy Software Development Kit for Web Developer Guide.*

3.2.3. MicroStrategy Intelligence Server Component

All MicroStrategy Intelligence servers are designed to generate SQL queries on-the-fly and execute them against a data warehouse via an Open Database Connectivity (ODBC) connection. However, some MicroStrategy implementations generate SQL against pre-built "Intelligence cubes," while other implementations join tables in an ad-hoc manner. The AQ system's Intelligence server is configured to generate SQL by joining database tables in an ad-hoc manner. In addition, the AQ Intelligence server is configured to receive requests and return result sets to the web server in XML format.

MicroStrategy's Intelligence Server provides an object-oriented architecture that drives SQL generation. All MicroStrategy projects on an Intelligence Server are made up of the same type of objects. Attributes, Facts, Metrics and Reports are all examples of object types that drive the generation of SQL. However, the specific implementation of these objects is unique for each AQ project. For example, the AQ SEDF project contains an Attribute object called "Age (10)," which is not present in the HDF project.

A report is the object "container" for all the objects needed to produce SQL queries and store results. Every user request generates an instance of a report object and that instance is passed from one MicroStrategy Intelligence Server sub-system to another as the report execution progresses.

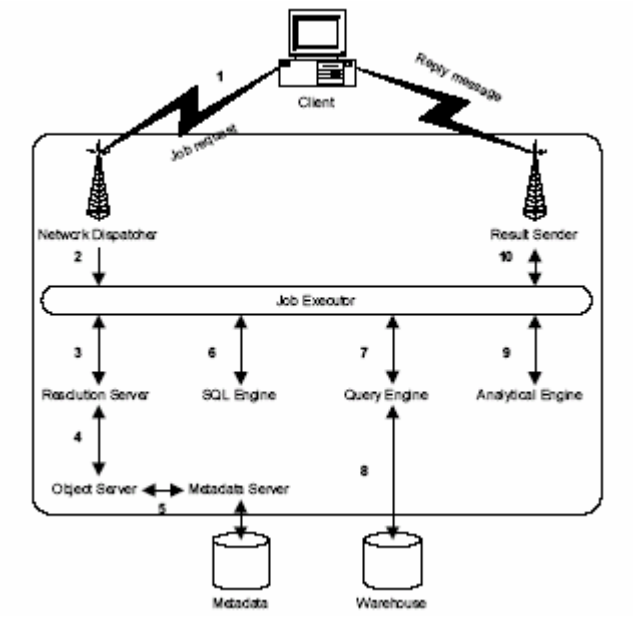
In order to enhance performance, the AQ system takes advantage of a **caching** mechanism provided by MicroStrategy to store Report object instances on the Intelligence server. This caching strategy is known by MicroStrategy as a "3-tier" environment. When a new user request is passed to the Intelligence server, the Intelligence server's Object Server checks the cache to see if the requested report already exists. If the report exists in the cache, the Intelligence server simply passes the result set contained in the report back to the Web Server and the process ends. If the report does not exist in the cache, the Intelligence Server builds a new Report object instance and executes SQL queries against the data warehouse.

It is important to note that the report cache can quickly consume disk space on the Intelligence Server. Although the report cache can be deleted manually from within the Intelligence Server Desktop console,

the report cache is deleted on a weekly basis when the Intelligence Server is rebooted (see section 9 System Maintenance).

Figure 3 and **Figure 4** below depict the execution flow of a report in the Intelligence server with and without caching enabled.

Figure 3: Report Execution Flow (no cache)



Source: MicroStrategy

1. The client application sends a request for a report

2. The Network Dispatcher receives the request and the Intelligence server checks user privileges.

3. The Job Executor sends the job to the Resolution Server

4. The Resolution Server checks to see if all the objects needed for the report are in memory. If not, it asks the Object Server for any missing object definitions.

5. The Object Server checks to see if there is a cached version of the report. If not, the Object Server queries the Metadata database to load object definitions in memory.

6. Once all objects are loaded, the Resolution Server returns the job to the Job Executor. The Job Executor sends the job to the SQL Engine for SQL generation. The SQL Engine generates the SQL and passes the job back to the Job Executor.

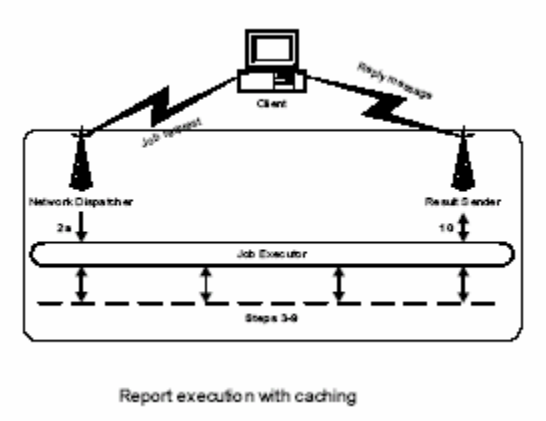
7. The Job Executor passes the job to the Query Engine, which submits the SQL to the data warehouse.

8. The Query Engine opens a connection to the warehouse and submits the SQL. After the SQL is executed and data is returned, the Query Engine passes the job back to the Job Executor.

9. The Analytical Engine processes the results and passes the job back to the Job Executor.

10. The Job Executor passes the job to the Result Sender, which sends the result set to the client.

Figure 4: Report Execution (cached report)



Source: MicroStrategy

1. The client application sends a request for a report

2. The Network Dispatcher receives the request and the Intelligence server checks user privileges.

3. The Job Executor sends the job to the Resolution Server

4. The Resolution Server checks to see if all the objects needed for the report are in memory and finding them, passes the request onto the Object Server.

5. The Object Server checks to see if there is a cached version of the report and finding one, passes it back to the Job Executor.

6-9. Skipped

10. The Job Executor passes the report instance to the Result Sender in a job and the Result Sender passes the result to the client.

For additional details on this topic:

- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the MicroStrategy Basic Setup Guide.*

- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the MicroStrategy Administrator, Intelligence Server, and Web Administrator Guide.*

3.2.4. Data Warehouse Component

The data warehouse component stores Census microdata. The data warehouse is a DB2 database server that consists of an HDF and SEDF database instance. Both databases were designed to optimize query execution time by implementing a STAR database schema. STAR schemas are highly de-normalized for the purpose of reducing the number of joins between related tables and thereby increasing performance.

3.2.5. MicroStrategy Metadata Database Component

The MicroStrategy metadata database contains Intelligence server project configuration information and object definitions. The metadata database consists of 10 tables, all of which are delivered by MicroStrategy and installed from MicroStrategy SQL scripts. No customizations were made to the metadata database schema, although the data contained within the database is unique to AQ and the Intelligence server cannot run without a connection to the metadata database.

4. APPLICATION DESIGN

4.1. MicroStrategy Customization Approach

Most AQ customizations were made to the user interface (UI) rather than the Intelligence Server. However, nearly every functional aspect of AQ is a combination of UI customizations and Intelligence server configurations designed to work in concert to produce an effect. The remainder of this document describes the customizations and configurations to MicroStrategy.

4.1.1. UI Customizations

The out-of-box installation of MicroStrategy Desktop on a web server includes the ASP code, stylesheets, JavaScript and images for a sample web site that integrates with a sample MicroStrategy Intelligence Server project. The HDF and SEDF web sites were developed from the sample, modifying the default MicroStrategy UI workflow and look-and-feel to meet the Census Bureau's business requirements. In addition to UI customizations, four stand-alone applications were written to generate XML navigation menus used in the AQ application UI and to produce usage reports from web log files. See **Table 1** below for a list of customizations made to the application code on the web server.

Table 1: Web Server Customizations

Component	Sub-Component	Description
Active Server Pages		
	cbCommonLib.asp	Common functions used in HDF and SEDF to evaluate objects
	cbFeedbackCuLib.asp	Processes user feedback and inserts it into feedback database
	cbHDFDataBrokerCuLib.asp / cbSEDFDataBrokerCuLib	Customized to use Census.dll to load DOM object references from XML files
	cbHDFGovernorCuLib.asp / cbSEDFGovernorCuLib.asp	Determines if the projected number of rows returned by a cross-tabulation is greater than what is allowed by the project configuration
	cbHDFNonQualCuLib.asp / cbSEDFNonQualCuLib.asp	Customized to auto-answer prompts in the non-qualifying report based on user selections currently in memory
	cbHDFPromptAlterXMLCuLib.asp / cbSEDFPromptAlterXMLCuLib.asp	Selects the list of valid derived measures
	cbHDFPromptInitializeCuLib.asp / cbSEDFPromptInitializeCuLib.asp	Customized to initialize the geogrpahic areas prompt based on the answers from the summary level prompt
	cbHDFPromptProcessCuLib.asp / cbSEDFPromptProcessCuLib.asp	Customized to auto-answer the population count report by using answers geographic and summary level prompts
	cbInboxCuLib.asp	Customized to get the saved tabulations, but does not allow access to other user's saved tabulations
	cbPreviewCuLib.asp	Customized to display a report header and footer for Census
	cbPromptCuLib.asp	Customized to alter a copy of the XML object in memory based on user response to a prompt
	cbPromptDisplayCuLib.asp	Customized to display a summary of the user selected prompts
	cbReportAttrFormsCuLib.asp	Customized to set the attributes for cross-tabulation based on user selections
	cbReportCuLib.asp	Customized to get the DSSID of the non-qualifying report to run
	cbReportsCuLib.asp	Customized to get the DSSID of the folder object where summary level report are stored
	cbReportSummaryCuLib.asp	Displays summary levels from XML file and retrieves user selected summary level to populate report prompt

Component	Sub-Component	Description
	cbTraceCuLib.asp	Customized to write application errors to HDF or SEDF log files rather than a single log file
	InboxCuLib.asp	Customized appearance of tabulations in queue
	MenuGUICuLib.asp	Customized the way drop-down menus appear
	ObjectBrowserGUICuLib.asp	Customized user preferences
	ObjectGUICuLib.asp	Customized user preferences
	PreviewCuLib.asp	Customized to get AQ specific POST/GET input parameters.
	PromptCuLib.asp	Gets user answers to prompts from memory
	PromptDisplayCuLib.asp	Customized to display a history of what attributes, universes and geogrpaphies a user selected to produce a report
	PromptProcessCuLib.asp	Gets user answers to prompts from memory
	ReportCuLib.asp	Displays a report
	SystemStatusCuLib.asp	Added to display system status message to users on login page
	/Admin/_toolbar_admin.asp	Creates a "system status" hyperlink on the Administration web site
	/Admin/SystemStatusEditor.asp	Provides the system status interface where administrators can create and send a message to users
	/Admin/SystemStatusEditorToolbar.asp	Provides an interface to create and send a message to users about status of system
JavaScript		
	CensusBureau.js	Opens a help topic window
	cbPromptFunctions.js	Add/Remove selections to list box
Images		
	/Images/Census	Census Bureau images
Help Files		
	/Help/Census	HTML and PDF files
Style Sheets		
	/style/CensusBureau.css	Styles used throughout user interface
Stand-alone Applications		
	HDFDataFiles.exe	A Visual Basic application that produces XML files for menu-driven options in the HDF project
	SEDFDataFiles.exe	A Visual Basic application that produces XML files for menu-driven options in the SEDF project
	Census.dll	A Visual C++ library that reads XML files into memory
	SRD.exe	A Visual Basic application that creates usage reports based on Web Log files.

4.1.2. Intelligence Server Customizations

Customizations to the out-of-box MicroStrategy Intelligence Server solution were minor. Two custom math functions, linear and pareto median, that were not supported by the out-of-box MicroStrategy product were added to the Intelligence Server as plug-ins (see **Table 2**). Most of the development on the Intelligence Server involved the creation and configuration of MicroStrategy objects using delivered MicroStrategy features.

Table 2: Intelligence Server Customizations

Component	Sub-Component	Description
Functions		
	BOCLMedian function	Custom function written in Visual C++, and installed on the Intelligence Server, to calculate a linear Median
	BOCPMedian2 function	Custom function written in Visual C++, and installed on the Intelligence Server, to calculate a Pareto Median

4.2. Data Security Design

In accordance with Title 13, Section 9 of the US code, the Census Bureau is obligated to protect survey respondents' identity. The Census Bureau and the AQ system both employ processes to prevent disclosure of individual information. All HDF and SEDF data is cleansed using various Bureau techniques prior to AQ receiving the data and loading it into the data warehouse. However, since it may still be possible to identify individuals from HDF and SEDF data, additional data security techniques are designed into the AQ system to protect the data, particularly from access by external users who generally do not have sworn status.

This section describes how the Census Bureau and the AQ system prevent data disclosure to unauthorized users.

4.2.1. Census Bureau Data Cleansing and Tabulation Policy

Although internal AQ users have access to data with relaxed disclosure restrictions, like external users, all the data they access has been subjected to Census Bureau data cleansing techniques. Data swapping is one technique used by the Census Bureau to cleanse HDF and SEDF data before providing it to AQ. The Bureau also may decide not to release certain data as a matter of tabulation policy regardless of cleansing. The Disability variable in AQ is an example of how tabulation policy limited disability data disclosure in AQ. Census survey respondents check one of 5 disability boxes on the long form. Bureau policy dictated that these 5 categories be collapsed into 2 categories in AQ.

- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the Census Summary File 1- Technical Documentation that describes data swapping and other data cleansing techniques.*

4.2.2. AQ Data Disclosure Protection

AQ uses cleansed data files from the Census Bureau as the building blocks for AQ. The extract, transfer and load process (ETL) developed for the AQ application categorized individual survey records into broad database categories that were migrated to the AQ data warehouse. Additional categories were created exclusively in the data warehouse based on the original categories to further hide microdata outliers. AQ obscures the information displayed to both internal and external users by hiding microdata outliers in broad database categories and displaying alternate values for outliers in order to make it difficult to identify respondents.

Because all AQ users are subject to these techniques, to some extent AQ hides raw microdata from internal and external users alike. However, external users are subject to additional data security techniques that not only obscure microdata records, but also hide the query results when certain

thresholds are reached. The techniques discussed below cover microdata obfuscation as well as data result filtering.

4.2.2.1. Microdata Obfuscation

Data obfuscation is achieved by configuring the attributes on the MicroStrategy Intelligence server to retrieve top-coded, bottom-coded and recoded demographic characteristics from data warehouse views and by configuring metric objects to use jam values.

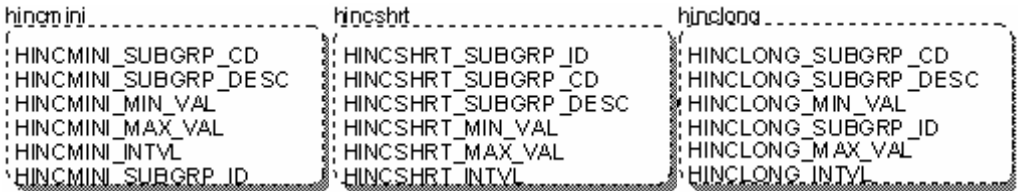
4.2.2.1.1 Top-coding and Bottom-Coding

Top-coding is a method of disclosure limitation in which all results in or above a certain percentage of the distribution are placed into a single category to hide outliers. All continuous variables such as household costs and income, are top coded to cap the maximum values displayed in AQ. For example, if the population count of household incomes is $150,000 (above $100,000), it would be displayed in a category called "$100,000 or more."

Similarly, bottom-coding is a method of disclosure limitation in which all results in or below certain percentage of the distribution are placed into a single category to hide outliers. For example, household incomes below $10,000 would be displayed in a category called "$1 or break even to $9,999 or less."

The data AQ receives from the Census Bureau is *not* top-coded or bottom-coded. The ETL process aggregates Census Bureau data into base categories that implement top and bottom coding. The AQ data warehouse expands top-coding and bottom-coding by implementing database views that aggregate ETL processed base tables into even larger categories. For example, the database views hincmini, hincshrt and hinclong aggregate the Household Total Income values stored in the hinc_val base table into categories that are top coded and bottom coded (see **Figure 5** below). The hincmini view exposes 8 categories of Household Total Income while hincshrt exposes 17 categories and hinclong exposes 40 categories.

Figure 5: Database views used for Top and Bottom-Coded Household Income



The Intelligence server attributes Household Total Income (8), Household Total Income (17), and Household Total Income (40) were created using the hincmini, hincshrt, and hinclong views as source tables. By using the hincmini view as it's source, the Household Total Income (8) attribute has access to 8 income categories with the bottom-coded category displayed as "*Less than $15,000*." Using the hincshrt view, Household Total Income (17) has access to 17 income categories with the bottom-coded category shown as "*Less than $10,000*." Using the hinclong view, Household Total Income (40)" has access to 40 income categories with the bottom-coded category displayed as "*Less than $2,500*."

For a complete list of top and bottom-coded attributes in AQ:

- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the Advanced Query User Guide – Census 2000 Hundred Percent Data.*
- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the Advanced Query User Guide – Census 2000 Sample Data.*

4.2.2.2. Recoding

Recoding is also a microdata obfuscation technique. Recoding is similar to top and bottom-coding where the intention is to hide outlying data points in larger categories. The difference is that instead of grouping just the extremely high and low values into new categories, all the data is re-categorized. It is entirely possible for a microdata statistic to include top-coding/bottom-coding as well as recoding. AQ variable

attributes such as race, age, occupation, industry, Hispanic origin and household relationship are recoded into predefined categories to show varying degrees of detail. .

For example, the data AQ receives from the Census Bureau contains population counts for 91 individual ages and is *not* recoded. AQ recodes ages by creating database views to aggregate specific ages into age range categories (i.e. 18 to 24 years). The table views: "agemini," "agemini4," "agebrief", "agevbrief", "ageshort," "agemidi," "agemed" and "agelong" are derived from the "age_val" table and implement top and bottom-coding as well as recoding for age. The MicroStrategy Intelligence Server is configured to use the age views listed above as source tables for its Age attributes: Age (3), Age (4), Age (9), Age (10), Age (14), Age (23), Age (38), and Age (91). These attributes are exposed to users on AQ's web site.

The Age (3) attribute, which uses the "agemini" view, contains 3 recoded age categories: "*Less than 18*," "*18 to 64*" and "*65 and over*." Notice that the first and last categories are bottom and top-coded.

A complete list of recoded attributes in AQ and the categories available within these attributes can be found in the HDF and SEDF Advanced Query User Guide:

- *See section* ***Error! Reference source not found. Error! Reference source not found.*** *for the pointer to the Advanced Query User Guide – Census 2000 Hundred Percent Data.*
- *See section* ***Error! Reference source not found. Error! Reference source not found.*** *for the pointer to the Advanced Query User Guide – Census 2000 Sample Data.*

4.2.2.3. Jam Values

Jam values are used to obscure calculated medians when those medians are statistical outliers. A jam value is a hard-coded value displayed to the user instead of a calculated value. In the AQ environment, jam values are displayed when median calculations are performed on top or bottom-coded categories. For example, if 24,602 households in Alabama fall into the recoded "$200,000 or more" Household Total Income category, the median income value displayed for this category will be $200,001 regardless of whether the true median is $200,001, $275,300 or $500,000. Likewise, the median value displayed for the Household Total Income category "Less than $10,000" will be"$9,999.0" instead of the true median (see **Figure 6**).

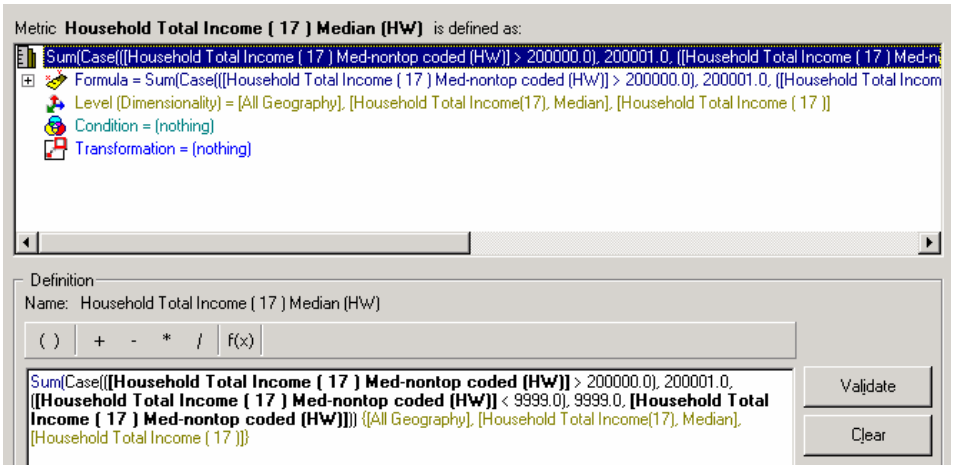
Figure 6: Jam value for Household Total Income (17) Median

State	Household Total Income (17)	Metrics	Count	Household Total Income (17) Median (HH)
Alabama	$200,000 or more		24,602	$ 200,001
	Total		24,602	
Total			24,602	

The use of jam values is essential for non-disclosure because recoding and top/bottom-coding merely groups the population count into categories (i.e. 200,000 or more), but these categories do not hide calculated statistics on the grouping.

Jam values are implemented in AQ by creating MicroStrategy metric object math formulas that indicate what value to display if the computed value is greater than or less than some value. **Figure 7** shows how the jam values for Household Total Income (17) are coded. The jam value in the figure is 200001 when the median is above 200000 and 9999 when the median is below 9999.

Figure 7: Jam values in the formula for Household Total Income (17) derived measure



In order to facilitate maintenance, derived measures with jam values were created in folder names corresponding to the attribute for which the derived measure applies. For example, the derived measure that calculates a mean value for Age (10) is located in the /Public Objects/Metrics/Derived Measures/Age(10)/Mean folder on the Intelligence Server and is completely separate from the /Public Objects/Metrics/Derived Measures/Age (14)/Mean derived measure.

For a list of SEDF and HDF metrics and their jam values:

- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the SEDF Median Jam Values.*
- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the HDF Median Jam Values.*

4.2.3. Filters

The jam value, top-coding, bottom-coding and recoding methods described earlier obfuscate data; they do not prevent cross-tabulations or hide the results of cross-tabulations. Filters do exactly that, they prevent a cross-tabulation from being selected as part of the tab definition or hide the results of cross-tabulations. Some filter rules affect both external and internal users, while others are specific to external users and enforce more stringent disclosure rules on external users.

Disclosure limitations are enforced with a pair of AQ filters: the **Query filter** and the **Results filter**. These filters are designed to suppress results for any geography that does not pass the filter criteria. In order to make the application code easy to migrate, the same code base is deployed on the web and Intelligence servers for external and internal AQ sites; however results filters are available but not imposed on the Internal site unless selected. Internal users who access the Intranet version of AQ can turn results filters on and off themselves via a radio button on the AQ website.

Filters can be turned on and off by editing an ASP configuration file; *CbHDFCommonDeclarations.asp* and *cbSEDFCommonDeclarations.asp* for the HDF and SEDF projects respectively. The *CB_ENVIRONMENT_USER* variable in these configuration files indicates whether filters are turned on (see **Table 3**).

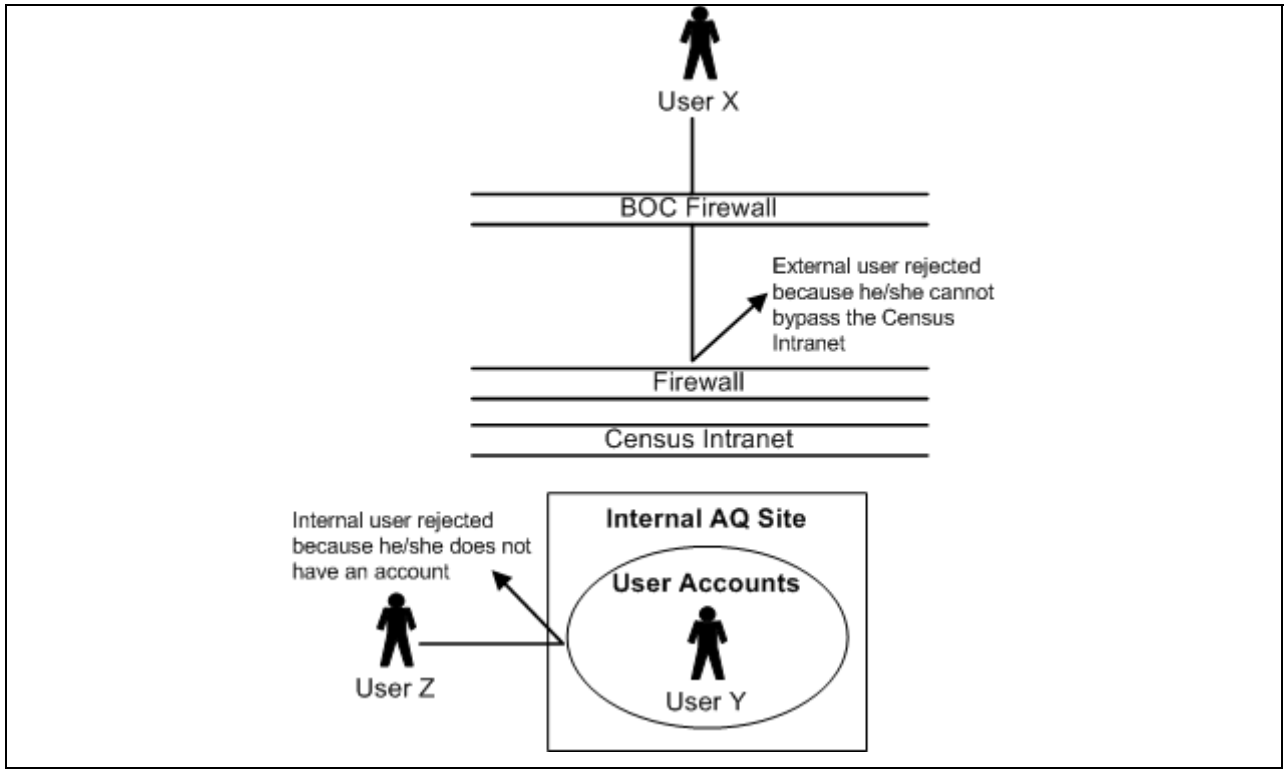
Table 3: Filter Settings

File	Variable	Value	Meaning
CbHDFCommonDeclarations.asp			
	CB_ENVIRONMENT_USER	1	Filtering off
	CB_ENVIRONMENT_USER	2	Filtering on
CbSEDFCommonDeclarations.asp			
	CB_ENVIRONMENT_USER	1	Filtering off

	CB_ENVIRONMENT_USER	2	Filtering on

It is not possible for external users to gain access to the internal site where filters are turned off unless they are on the Census Bureau Intranet and the AQ Administrator has created a login for them on the internal Intelligence server (see **Figure 8**).

Figure 8: Census Intranet Access Attempt



4.2.3.1. Query Filter

The query filter is designed to detect those queries that will not pass disclosure limitations before they are submitted for execution. This saves database resources and optimizes user experience by quickly identifying cross-tabulation options that would result in data the user is not entitled to view. For example, if an external user tries to add the geography with a population of 200 or less to the cross-tabulation, a pop-up window will alert the user that geographic areas with a population less than 200 cannot be added to a tabulation. The query filter enforces 7 disclosure rules. The first five query filter rules limit the type of data accessible to external and internal users. The last three filter rules only affect external users.

1. Cross-tabulations can only be created from the Census Bureau's predefined list of universes, geographies and attributes.
2. The smallest geography selection allowed in HDF is a block group, while the smallest geography selection allowed in SEDF is a census tract.
3. Derived measures such as mean and median will be displayed only for a few designated attributes.
4. Derived measures such as mean and median are provided for an attribute only if corresponding counts are available.
5. Some recoded variables are not allowed for inclusion in a cross-tabulation based on the population size in the selected geography. For example, the Age (91) category, with 91 recoded age ranges, is only available for inclusion in a cross-tabulation if the selected geography has a population of 100,000 or more. The query filter recognizes three population thresholds (A, B, C). Recoded variables are assigned to one or more of these population thresholds:
    - A – Less than 4,300 (small)
    - B – 4,300 – 99,999 (medium)
    - C – 100,000 or more (large)
6. The maximum number of variables that can be added to a cross-tabulation is three.
7. Cross-tabulations cannot include geographies with a population less than 200.

Query filter rules are implemented using a combination of JavaScript, ASP, and XML on the Web Server and are custom coded rather than out-of-the box features of MicroStrategy.

Rules 1-4 limit the navigation and drill down options available to users, thereby indirectly limiting the data users can access. Rules 1-4 are enforced by listing allowed universes, geographies and derived measures in five XML files stored on the web server (see **Table 4**).

Table 4: Query Filter XML Files

File Name	Function
SummaryLevelFilters.xml	Lists geography names that can be reached from the geography filter
SummaryLevelReports.xml	Identifies the available geographies
TemplateHierarchyMap.xml	Identifies the hierarchical relationship between geographies
UniverseFilters.xml	Identifies the Universes that can be displayed
DerivedMeasures.xml	Identifies the attributes that will display mean and median derived measures

The HDF and SEDF projects have their own version of these 5 files in the *DataFiles/HDF* and *DataFiles/SEDF* folder on the web server, making it possible to allow different geographies and derived measures to appear in the HDF and SEDF projects. Although the geographies and derived measures are listed in these XML files, they are updated using a Visual Basic application that inspects the Intelligence server hierarchy and metrics to produce the XML files. Therefore, these configuration files only reflect what is in the Intelligence server and are created by extracting information from the Intelligence server. The decision to extract navigation information from the Intelligence server into XML files rather than query it each time a user logged in was driven by the desire to improve performance for menu-driven activities that rarely change between releases. A DLL installed on the Web Server as

Microsoft Transaction Server Package, opens and reads the XML into the ASP pages. Off-loading constant parsing of XML by the ASP pages makes the display more efficient since Active Server Pages take much longer to parse XML.

For details on how to run the SEDFDataFiles and HDFDataFiles Visual Basic tool used to build XML or to install the DLL that reads the XML:

- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the AQ Deployment Procedures.*

Rule 5 limiting use of recoded categories is enforced in a series of XML files generated by the SEDFDataFiles and HDFDataFiles tool. The XML files generated by the tool list all the recoded attributes that are safe to use in a particular universe at a specific population level. A set of these files is deployed separately for the HDF and SEDF projects on the web server under *DataFiles/HDF/UnivAttr* and *DataFiles/SEDF/UnivAttr* folders.

When the user selects a universe and a number of geographies, the AQ application code determines the geography with the smallest population count and then appends two letters to the end of the Universe's object id. These two letters depict the parent universe and the population size.

The parent universe Identifiers are:

1. All People – A
2. Households and Housing Units – B
3. People in Households – C

The population threshold identifiers are:

1. Minimum population of 200 – A
2. Minimum population 4,300 - B
3. Minimum population 100,000 or greater – C

For example, if a user selects all females in Eureka County, Nevada (pop 1,695), the All Females universe object id B95A511C4AE63B8057DAEDA0345D59FC becomes B95A511C4AE63B8057DAEDA0345D59FC**AA** and points to B95A511C4AE63B8057DAEDA0345D59FC**AA**.xml in the DataFiles/SEDF/univAttr web server folder. The first appended letter "A" indicates that the parent universe of "All Females" is "All People." The second "A" indicates that recodes for the smallest population threshold should be used. For Internal users, the ASP code always selects the XML file with the broadest list of recodes (objectid**AC**, objectid**BC** or objectid**CC**), effectively nullifying rule 5 for internal users. The function cbSEDFGetUserQualifiedListID(strListID) in cbSEDFPromptInitializeCuLib.asp switches rule 5 on or off based on the CB_ENVIRONMENT_USER variable.

All AQ Universe names are listed UniverseFilters.xml where every universe has an object id that uniquely identifies that universe in the MicroStrategy metadata database.

Rule 6 is implemented entirely through JavaScript. The maximum variable count is hard coded in the HTML source for external users: <input type="hidden" name="cbDisclosureMaxVariables" value="3" />. The JavaScript function cbAddVariable uses this hidden field to prevent external users from adding more than 3 variables to a cross-tabulation. Limiting the number of variables in the "where" clause of a query protects confidentiality by leading to less sparse groupings in the tabulation results.

Rule 7 is also implemented with JavaScript. The minimum allowable population count for a geography is hard coded in the HTML source for external users: <input type="hidden" name="cbDisclosureMinPopCount" value="200" />. When an external user attempts to add a geography to the cross tabulation, the JavaScript function cbDisclosureMinPopCount compares the cbDisclosureMinPopCount to the population count of the geography by stripping out the geographic population count embedded in the name displayed to the user (i.e. Alabama (pop. 4,447,100)). Restricting users to larger populations protects confidentiality by leading to less sparse groupings in the tabulation results.

Additional details can be found in the HDF and SEDF Advanced Query User Guide.

- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the MicroStrategy Advanced Query User Guide – Census 2000 Hundred Percent Data.*
- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the MicroStrategy Advanced Query User Guide – Census 2000 Sample Data.*

4.2.3.2. Results Filter

The results filter processes data that has passed the query filter's criteria. It is designed to execute the selected cross-tabulation against the data warehouse, and then remove any single geography whose results fall below a specified mean, median or sparsity ratio threshold. The underlying philosophy of the design is that tabulations will be released if, and only if, they pass all of the rules. The tabulation results are never changed through rounding or other techniques to add noise to the results.

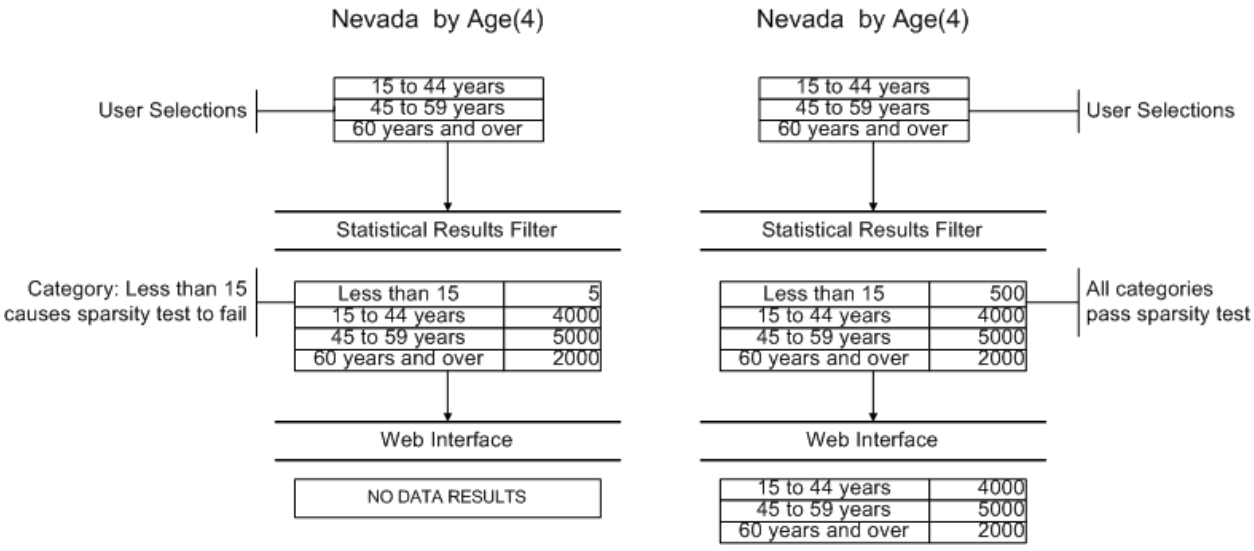
The results filter is actually represented by two different Filter objects in MicroStrategy, a housing filter object and a population filter object. References to the results filter include the housing and the population filter. The housing and population filters operate identically except that the population filter is used for the "People" universe and the housing filter is used for queries against the "Households and Housing Units" universe and "People in Households" universe.

The results filter computes results using a two-stage process. In the first stage, the results are retrieved as if no restrictions had been specified for demographic variables. For example, in a cross-tabulation of Nevada by Age (4), if a user selects only 3 of the 4 categories, the results filter calculates mean cell size, median cell size and sparsity ratio on all four categories as if the user had not restricted the Age (4) demographic variable. If the 4th category causes mean, median or sparsity to fail the filter, all the results in this geographic sub-table are suppressed (see **Figure 9**).

By calculating statistics against all attribute categories, the results filter is designed to prevent complementary disclosure. Complementary disclosure occurs when results for three out of four categories in a demographic variable like Age (4) are displayed, allowing the user to calculate the fourth category directly or by running a second query with Age (10) that does pass the filter.

In the second stage of filter processing, if results pass confidentiality tests, then the table is narrowed to return only the original demographic categories specified by the user (see **Figure 9**).

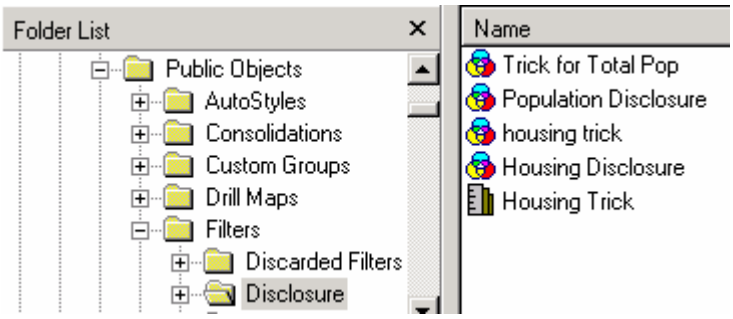
Figure 9: Results filter - The effect of categories on results



In the SEDF project, the result filter is configured so that any geography whose mean, median or sparsity falls below disclosure review board specified values will not be displayed in the results (see Advanced Query Confidential Appendix). In the HDF project any geography whose mean, median and sparsity falls below disclosure review board specified values will not be returned to the user in the results (see Advanced Query Confidential Appendix).

The results filter is implemented on the MicroStrategy Intelligence server as a Filter Object that is added to the Report Object and used during the processing of the report. As previously mentioned, there are two Filter Objects in the SEDF and HDF project, the Population Disclosure Filter and the Housing Disclosure Filter. The Population Disclosure Filter is used within the "All People" universe, while the Housing Disclosure Filter is used within the "Households and Housing Units " and "People in Households" universes. **Figure 10** below depicts the two filter objects.

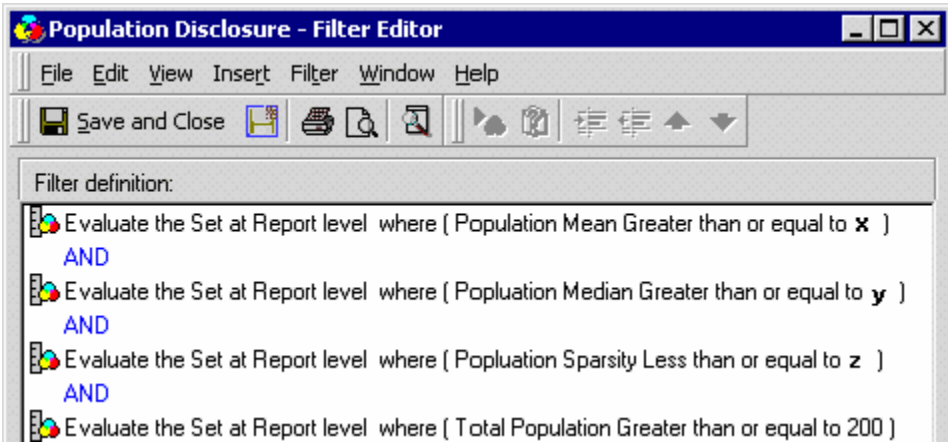
Figure 10: Population and Housing Filters



Both Filter objects are configured to calculate mean, median and sparsity by using the housing and population metrics under *Public Objects\Metrics\Disclosure Metrics.* The mean and median used for determining disclosure is different than the mean and median derived measure calculations displayed to users when they select an attribute for cross-tabulation. Not only are the derived measure mean and medians represented by different metric objects than the disclosure filters, but also, they calculate mean and median using a different formula (see section 4.3.5.1).

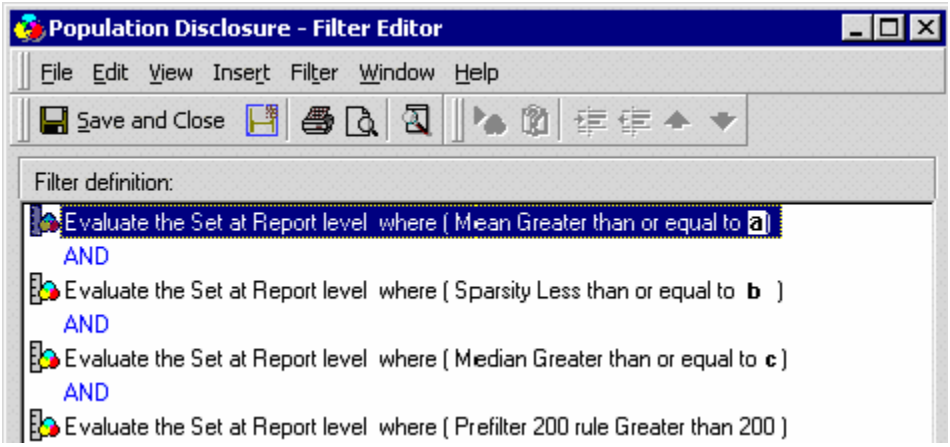
By clicking on the housing or the population filter object on the Intelligence server, it's possible to see the mean, median and sparsity thresholds that were configured by the application developer (see **Figure 11** and **Figure 12**).

Figure 11: SEDF - Configuration of Results Filter Thresholds for Population



[T]Substitution values for variables are in the Advanced Query Confidential Appendix

Figure 12: HDF - Configuration of Results Filter Thresholds for Population



[T] Substitution values for variables are in the Advanced Query Confidential Appendix

4.2.3.2.1 Disclosure Mean

MicroStrategy filter objects configure mean, median and sparsity limits, but metric objects on the Intelligence server are responsible for calculating mean, median and sparsity values. Typically, a mean is defined as the total population divided in half. However, in accordance with direction from the Census Bureau, the AQ mean filter calculates mean using the ratio of total population in a geographic sub-table to the total number of matrix cells in the sub-table (see **Figure 13**). A geographic sub-table is the matrix of cells produced by all combinations of the attributes selected for cross-tabulation in a geography. For example, a cross-tabulation of two counties by Disability (3) yields two sub-tables, one for each county. Each sub-table will have 3 matrix cells, one for each category of disability (see **Figure 14**). Since the Census Bureau's definition of mean does not correspond to the out-of-box mean function provided by MicroStrategy, it was necessary to configure the mean metric objects as a composition of metric objects that use standard division to calculate a mean (see **Figure 15**).

Figure 13: Mean formula

$\frac{t}{z}$ where **t** is a geographic sub-table's population count and **z** is the number of matrix cells in the sub-table

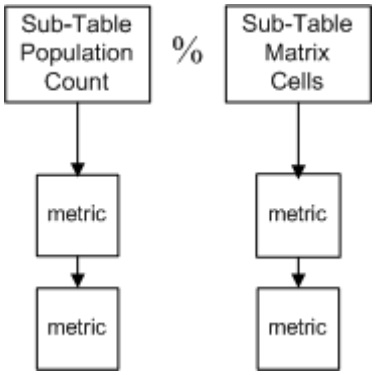
[T] Substitution values for variables are in the Advanced Query Confidential Appendix

Figure 14: Example of one geographic sub-table

	Attribute: Disability (3)	
Geo: Lewis, County, Alabama	Under 16, Not in Universe	0
	Disabled	2000
	No Disability	3000
		t=5000

Z=3

Figure 15: Mean metric object configuration using standard division function



When the mean falls below a specified threshold, the geography sub-table is removed from the results displayed to users. Mean is likely to fall below the permissible threshold if the sub-table population is spread across many matrix cells or if the sub-table contains many zero-filled cells (not applicable) for a cross-tabulation.

In SEDF every individual is weighted so that the total population counts in SEDF match those in HDF. For example, a single person in SEDF might be shown as 30 people by applying a multiplier of 30 as a weight. Census Bureau weighting techniques are beyond the scope of this documentation, however, the presence of weighted data has an effect on the design of filters in SEDF vs. HDF. In SEDF, the weighted totals are used to calculate mean; so the mean filter must be higher than in HDF to protect against discloser of what really represents a single individual.

4.2.3.2.2 Disclosure Median

The results filter calculates the median value of a population for a geographic sub-table to determine if the median falls below a configured threshold. The filter uses the MicroStrategy delivered median function to calculate the median value for a cross-tabulation (see section 4.3.5).

A median is sometimes thought of as the middle value in a data set and is the standard way that medians are calculated. In order to calculate a median, data must first be sorted in ascending order. The median is the value at the middle of the distribution as shown in the formula in **Figure 16**. If the median data point is 4.5; the median will default to the 4th data point. If the median data point is 4.6; the median calculation will select the 5th data point.

Figure 16: Linear Median formula

$$\frac{n+1}{2}$$ where **n** is the number of values in a set of data

[1] Substitution values for variables are in the Advanced Query Confidential Appendix

4.2.3.2.3 Disclosure Sparsity Ratio

The results filter calculates the sparsity ratio to determine if the sparsity exceeds a configured threshold. Since there is no MicroStrategy sparsity function, the Metric Object used to calculate sparsity is configured to use a compound formula that computes the ratio (see section 4.3.5). The formula and application for sparsity in the HDF and SEDF project is fundamentally different. Not only are the sparsity thresholds different (**c** and **z**), the formula used to calculate sparsity is different (see **Figure 17** and **Figure 18** below).

Figure 17: Sparsity formula for HDF

$\frac{m}{t} > c$ where **m** is number of cells with a population count=1 and **t** is the number of nonzero cells

[1]Substitution values for variables are in the Advanced Query Confidential Appendix

Figure 18: Sparsity formula for SEDF

$\frac{n}{v} > z$ where **n** is number of cells with a population count less than **s** and **v** is number of nonzero cells

[1]Substitution values for variables are in the Advanced Query Confidential Appendix

The Disclosure Review Board intended for HDF and SEDF sparsity tests to reveal the same amount of data. The formulas to achieve the same results are slightly different because SEDF is weighted and HDF is not weighted. In SEDF, cells with a population count less than s, a weighting factor, are used to determine sparsity. In HDF, cells with a population count of 1 are used to determine sparsity.

Please refer to the following documentation for examples for calculating mean, median and sparsity in the disclosure filters:

- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the Example of a Mean Calculation document.*
- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the Linear Interpolation Median Calculation Example document.*
- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the Example of a Sparsity Calculation document.*

4.3. Schema Object Model Configuration

A Schema Object Model consists of all the objects used to define a MicroStrategy project. The data security section of this document described some objects like attributes and metrics and how they are configured to enforce data security policies. However, additional configurations were made to meet other design goals. The discussion below outlines the types of objects in the SEDF and HDF project as well as how they were configured to meet specific AQ design goals.

4.3.1. Attribute Objects

An attribute is a column in one or more lookup tables in the data warehouse. An attribute provides a handle for both aggregation and filtering, and it represents a level of aggregation. As discussed in section 4.2, attributes in AQ represent demographic characteristics like age and often reflect lookup tables that have been recoded into categories to protect confidentiality. Attributes in AQ are usually sourced from views, but some come directly from tables in the data warehouse. Attribute names were created on the fly on the Intelligence server and are not related to any database table in the data warehouse. In order to improve performance the MicroStrategy Intelligence Server is configured to cache data warehouse tables in memory without having to query the warehouse for these elements in subsequent requests. Due to the fact that demographic totals for demographic characteristics like age and race sometimes change when the Census Bureau releases updates and corrections, these attributes were not configured to use "element caching." However, geographic attributes in AQ do not change and therefore, the design team turned "element" caching on for geographic attributes.

4.3.2. Fact Objects

Fact objects relate numeric data in the data warehouse to the user, using physical columns from the data warehouse. Some AQ facts exist to aggregate population counts for a demographic characteristic, while others are workarounds to allow metrics to perform a calculation. An example of a workaround fact is the

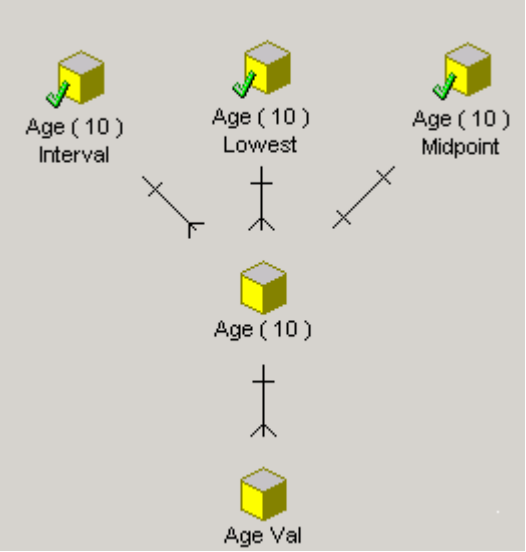
so-called Zero fact. The zero fact was created in AQ to count the number of cells in a cross-tabulation that will have no response. The zero fact is used to support filter calculations, but is not used to display any data to the user.

4.3.3. Parent-Child relationships

MicroStrategy typically requires a project to define parent-child relationships between attributes in order for the Intelligence server to instruct its SQL Generation Engine to create a join between the tables referenced by the parent and child. However, the Advanced Query application also uses parent-child relationships as a workaround to permit data aggregation on both the child and parent values.

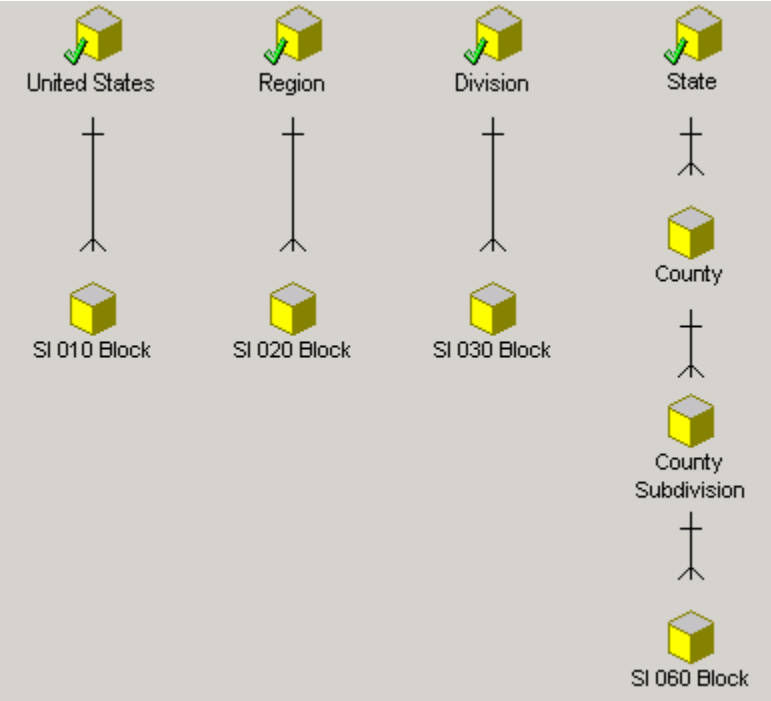
In order for data to be aggregated in MicroStrategy, the data must be linked to an attribute, rather than a "form" of the attribute. The attribute for Age (10) has related attributes for the age midpoint, lowest internal and highest Interval. These related attributes are modeled as separate attributes rather than "forms" so that age data can be aggregated and have calculations on midpoint, lowest and intervals. For example, Midpoint is the parent of Age (10) and Age (10) is parent of Age Val (see **Figure 19**). This is important because the midpoints for each age (i.e. 76.5 for 76) for are used to calculate derived measure mean values. The lowest and highest interval for each age contribute to figuring out the derived measure median,

Figure 19: Example parent-child relationships for demographic attributes



Standard parent-child relationships are implemented in AQ for geographic attributes. For example, state is a parent of the county attribute (see **Figure 20**). The purpose of the geographic hierarchies created by the parent-child relationships is to indicate to the SQL generator that the parent should be joined to all the relevant child attributes where population counts are stored. For example, all the blocks of state need to be identified in order for a population count to be tallied. Note that even though the smallest geography users are allowed to select in SEDF is a Census Tract, the underlying application still uses blocks to store and retrieve demographic statistics.

Figure 20: Example parent-child relationships for geographies in SEDF



The parent-child design for geographies works well with so-called **Natural Hierarchies**. Natural hierarchies are geographies that completely encompass another geography. For example, counties do not cross state boundaries. However, as part of the AQ design, the application also needs to model geographies that cross multiple boundaries or so-called **Cross Tabulation Areas**. For example, in the HDF project, a Voting District can occur as part of a state or county. Cross tabulation areas were also implemented in AQ using parent-child relationships, but with multiple parents.

4.3.4. Hierarchy Objects

Although parent-child relationships create implicit hierarchies, there is a special object in MicroStrategy called a "hierarchy." In addition to defining parent-child relationships, objects that are members of a MicroStrategy hierarchy can be used as prompts. For example, a user can be prompted to select a geography in a hierarchy and then continue to drill down to related geographic locations.

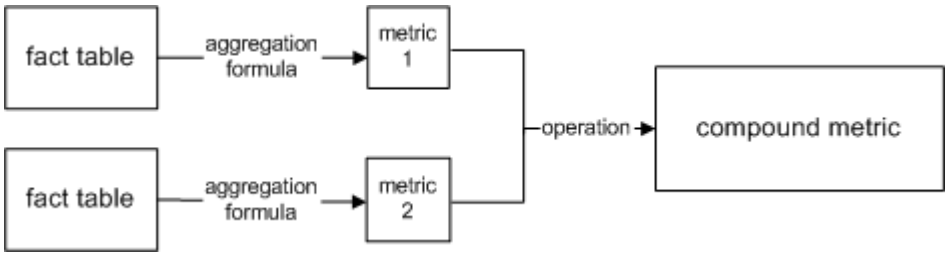
AQ geographies and universes are modeled as hierarchies. AQ Universes such as "All People" contain sub-universes like "All Males, All People" and "All Females, All People." Since the AQ design requires users to be prompted to select a universe and then child universes, universes need to be implemented as hierarchies. Most of the universes in AQ are routinely used by the Census Bureau in other applications and reports; however a few universes are unique to AQ.

The decision to create universes that exist only in the AQ environment was driven by the desire to expose more cross-tabulation options to users. For example, in AQ there is an "All Males, All People" universe as well as a Sex (2) variable that gives users the option to refine a cross-tabulation to all males or all females. By creating an "All Males, All People" universe the design team effectively eliminated the need for a user to "waste" one of their three cross-tabulation variables on gender specification. In addition, it was discovered that certain universe and variable combinations produce meaningless results in AQ queries. For example, It doesn't make any sense to allow users to query the "People Less than 5 years" universe by "Class of Worker" since we know that children in the US are not employed. To prevent the user from specifying these combinations, some variables will not appear on the selection list if a particular universe is selected. The absence of a variable in a specific universe is controlled by removing the attribute from the universe's hierarchy.

4.3.5. Metric Objects

Metric Objects define calculations to be performed against facts to produce a metric value. The AQ application defines several groups of metrics 1) disclosure metrics 2) nondisclosure metrics and 3) derived measure metrics. All metrics in AQ are compound metrics, metrics that are defined by other nested metrics. Creating compound metrics increases the maintainability of the code by keeping each part of the metric equation atomic (see **Figure 21**). When a compound metric is updated, the entire metric formula is also updated to reflect the modification. All AQ metric objects are configured to perform and inner join between the fact tables if they are composite metrics. The metric formatting feature is used extensively to format metric results as general numbers or currency.

Figure 21: Compound metric



Disclosure, Non-Disclosure and Derived Measure Metrics are configured on the Intelligence server in the locations indicated in **Table 5**, **Table 6**, and **Table 7** below.

Table 5: Location of Disclosure Metrics on Intelligence Server

Project	Disclosure Metrics	Location
HDF		
	Mean	\Metrics\Disclosure\Mean Cell Size
	Median	\Metrics\Disclosure\Median Cell Count
	Sparsity	\Metrics\Disclosure\Sparsity Count
SEDF		
	Mean	\Metrics\Disclosure Metrics\Mean
	Median	\Metrics\Disclosure Metrics\Median
	Sparsity	\Metrics\Disclosure Metrics\Sparsity

Table 6: Location of Non-Disclosure metrics on Intelligence Server

Project	Non-Disclosure Metrics	Location
HDF		
	Mean	\Metrics\Nonqualifying Metrics
	Median	\Metrics\Nonqualifying Metrics\Median Nonqualifying
	Sparsity	\Metrics\Nonqualifying Metrics\Sparsity Ratio
SEDF		
	Mean	\Metrics\NonDisclosure Metrics\Mean
	Median	\Metrics\NonDisclosure Metrics\Median
	Sparsity	\Metrics\NonDisclosure Metrics\Sparsity

Table 7: Location of Derived Measure metrics on Intelligence Server

Project	Derived Measure Metrics	Location
HDF		
	Mean	\Metrics\Derived Measures
	Median	\Metrics\Derived Measures
SEDF		
	Mean	\Metrics\Derived Measures
	Median	\Metrics\Derived Measures

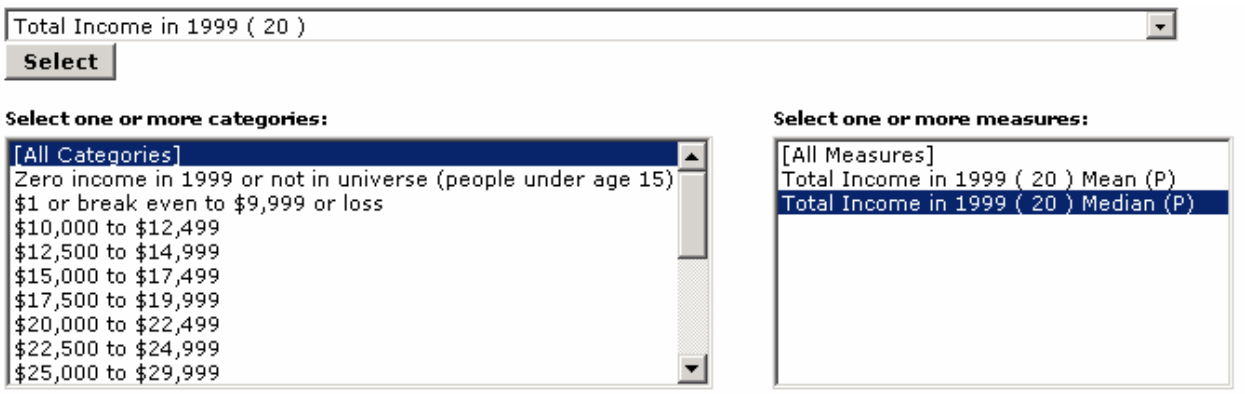
The disclosure metrics are designed to calculate the mean, median and sparsity of a geography using the formulas described in section 4.2.3.2. The disclosure metric is used to filter out geographies from results. The non-disclosure metrics also calculates the mean, median and sparsity of a geogpraphy, but is used to report to the user what geographies were filtered and why. The difference between the disclosure and non-disclosure metric is in the metric formula. The formula used in the non-disclosure report is configured to return a 1 or 0 value indicating a pass-fail status that the user can view. The formula in the disclosure metric returns a mean, median and sparsity value that is not viewed by the user. Since there are separate fact tables to count people and housing units, it was necessary to configure mean, median and sparsity metrics for the People and Housing universes in HDF and SEDF; thus each project has 6 disclosure and non-disclosure metrics.

Like disclosure and non-disclosure metrics, derived measure metrics also calculate a mean and median. However unlike disclosure and non-disclosure metrics, these measures are not calculated against the raw data. Instead derived measures are calculated from the filtered results, leading to vastly different results and disclosure issues. The discussion that follows describes how and why derived measures were configured to use other statistical methods to calculate a mean and median displayed to users.

4.3.5.1. Derived Measures

For some select attributes, the Census Bureau elected to make mean and median calculations available to the user for inclusion in the cross-tabulation. For example, see **Figure 22**.

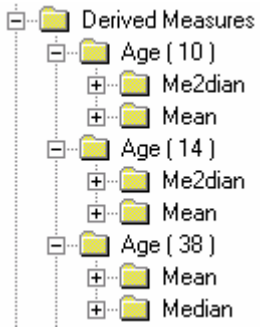
Figure 22: Derived Measures in AQ



All derived measures are placed in folders under the attribute name for which they apply (see

Figure 23). The ASP code only recognizes and displays derived measures in folders named “Mean” and “Median.” Occasionally, folders were renamed “Me2dian” or “M2ean,” to prevent the derived measure from appearing on the AQ web site without removing the metric from the Object Schema.

Figure 23: Derived Measure Folder Naming Conventions



4.3.5.1.1 Mean

The mean calculation in the disclosure and non-disclosure metrics for the results filter is a calculation that is used to prevent too many cells from displaying single for few instances. The mean calculation for derived measures is fundamentally different. It provides an average for the result set by multiplying the population count in a category by the midpoint for the category that is listed in the data warehouse. The products are then summed and divided by the total population as shown in **Figure 24**.

Figure 24: Mean formula for derived measures

$$\frac{\sum n_i \times m_i}{\sum n_i}$$

where **n** in the numerator is the number of people or houses observed in an attribute category, **m** is the midpoint; and **n** in the denominator is total number of people or houses observed in all categories of an attribute

4.3.5.1.2 Linear and Pareto Median

The purpose of the median calculation performed for the results filter is to suppress results where the population counts are generally low. The purpose of the derived measure median calculations is to display a median value to users derived from results that have passed the filters. Since the derived measure median is not calculated against the raw data, the application does not need to re-execute a query to get the entire table of results. This saves database resources and speeds response time for users who elect to include median in their results. However, performing median calculations on categories requires special statistical techniques in order to generate a median. Depending on the attribute, derived measure medians are calculated using a linear median or what is known as a Pareto median.

A linear median is the middle value in an ascending sequence of values of a variable. As directed by the Census Bureau, linear medians are appropriate for all but income variables category intervals greater than $2,500. Thus, the median for the population distribution by age (10) is calculated in AQ using a linear median interpolation between the upper and lower bounds of the interval in which the midpoint count is located.

In Pareto interpolation, the median is derived by interpolating between the logarithms of the upper and lower income limits of the median category. It is used by the Census Bureau in calculating median income within intervals wider than $2,500.

Linear and Pareto medians are not standard functions included in MicroStrategy. A custom function called, **BOCLMedian**, was written in Visual C++ to calculate Linear medians on all attributes that don’t

qualify for Pareto interpolation. A custom function, called **BOCPMedian2** was written in Visual C++ to calculate Pareto medians on recoded income and earnings attributes with category intervals of $2,500 or more. It's possible to determine which type of median an attribute is using simply by examining the name of the median function in the derived measure formula. The BOCLMedian and BOCPMedian2 custom functions are installed on the Intelligence server under *Schema Objects\Functions and Operators\Plug-In Packages\BOCFunctions*

For details on the formula for Linear Median:

- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the Example of a Linear Interpolation Median Calculation document.*

For details on the formula for Pareto Median:

- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the Pareto Interpolation document.*

4.3.6. Disclosure and Non-Disclosure Report Objects

Two MicroStrategy report objects were created for the AQ application, a "disclosure" report and a so-called "non-disclosure" report. The results of a query are displayed in a disclosure report object. From a hyperlink in the disclosure report, users can view the non-disclosure report. A non-disclosure report lists any geographies that were suppressed from the results in the disclosure report due to failing the mean, median or sparsity limitations in the results filter.

4.3.6.1. Disclosure Report

A disclosure report was created for each summary level in the HDF and SEDF projects. **Table 8** identifies the location of the disclosure reports on the Intelligence server and provides a partial list of their names.

Table 8: Partial list of disclosure reports and locations

Report Name	Location	
Hundred Percent Data Tabulation – County	HDF Project	Public Objects\Reports\Disclosure Reports\Summary Level Reports
Hundred Percent Data Tabulation – Block		
Sample Data Tabulation – County	SEDF Project	Public Objects\Reports\Summary Level Reports
Sample Data Tabulation - Block Group {in Tract in County}		

The decision to create a report for each summary level was driven by the way data is stored in the data warehouse. Housing and population counts are stored by geography. In order to instruct the disclosure report to display a geographic location and the population count in that location, attributes and metrics must be included in the report.

The results displayed on the disclosure report are the housing and population counts; and in some cases, derived measures, if they were added to the cross-tabulation. For the external user base, the disclosure report is processed further by using disclosure metrics to suppress geographic locations which exceed mean, median and sparsity thresholds. The ASP variable that configures AQ for use by external users, also tells the application to pass the disclosure metrics into the disclosure report for external users only.

See section 4.2.3.2 for the formulas used in the disclosure metrics to filter geographies

4.3.6.2. Non-Disclosure Report

A total of 4 non-disclosure report objects were created in AQ to accommodate the HDF and SEDF projects as well as the people and housing universes within those projects. In the HDFproject's non-disclosure reports were bundled into the same folder as disclosure reports. Also, the non-disclosure report for people is not located in its own subfolder like Housing.

Table 9 below, identifies the location of each non-disclosure report on the Intelligence server. In the HDFproject's non-disclosure reports were bundled into the same folder as disclosure reports. Also, the non-disclosure report for people is not located in its own subfolder like Housing.

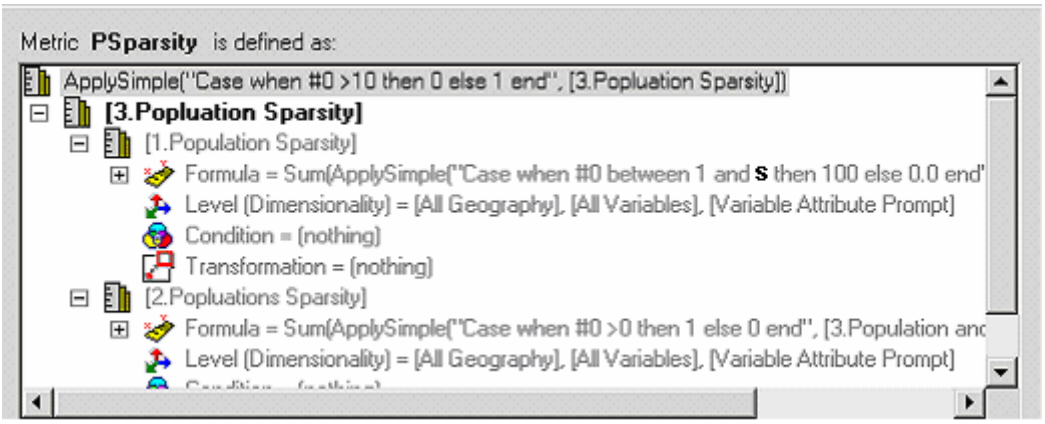
Table 9: Non-Disclosure report locations

Report Name	Location	
NonDisclosure	HDF Project	Public Objects\Reports\Disclosure Reports\Housing\NonDisclosure Public Objects\Reports\Disclosure Reports\NonDisclosure
	SEDF Project	Project Objects\Reports\Housing\NonDisclosure Project Objects\Reports\NonDisclosure

Since projects are independent of each other, it was necessary to create a non-disclosure report for each project. In addition, it was necessary to design a separate report for the people and housing universes since the reports use non-disclosure metrics and the non-disclosure metric is different for people and housing.

As mentioned in section 4.2.3.2, the formulas for the non-disclosure metrics are the same as the disclosure metrics used to suppress data from results, except the non-disclosure formulas return a 1 or 0 pass-fail value rather than the actual mean, median or sparsity ratio (see **Figure 25** below). Since the return value of the mean, median and sparsity ratio is different it was necessary to create metrics specifically for the non-disclosure report.

Figure 25: Example sparsity formula for non-disclosure report



[T]Substitution values for variables are in the Advanced Query Confidential Appendix

5. SYSTEM SECURITY

All AQ web servers, intelligence servers and database servers are hosted in the Census Bureau's Bowie, Maryland hosting center. All AQ hardware sites on the Census LAN are accessed through a firewall managed by Census Bureau Telecommunications office (TCO). The AQ Administration Team (IBM) maintains and manages the AQ hardware.

5.1. AQ Network Security Architecture

When discussing AQ Network Security, it is important to distinguish between the production, product assurance and development environments. The product assurance (PA) and development environments are hosted on the Census LAN, protecting their web servers, Intelligence servers and database servers from public access. However, the production **Internet** filtered AQ site is exposed to the Internet and therefore special demilitarized zone (DMZ) network segregation techniques were used to protect it. The production **intranet** unfiltered AQ site is largely hosted on the Census Bureau LAN, however, to conserve resources, it accesses the same database server used by the Internet AQ site in a DMZ (see **Figure 26** below).

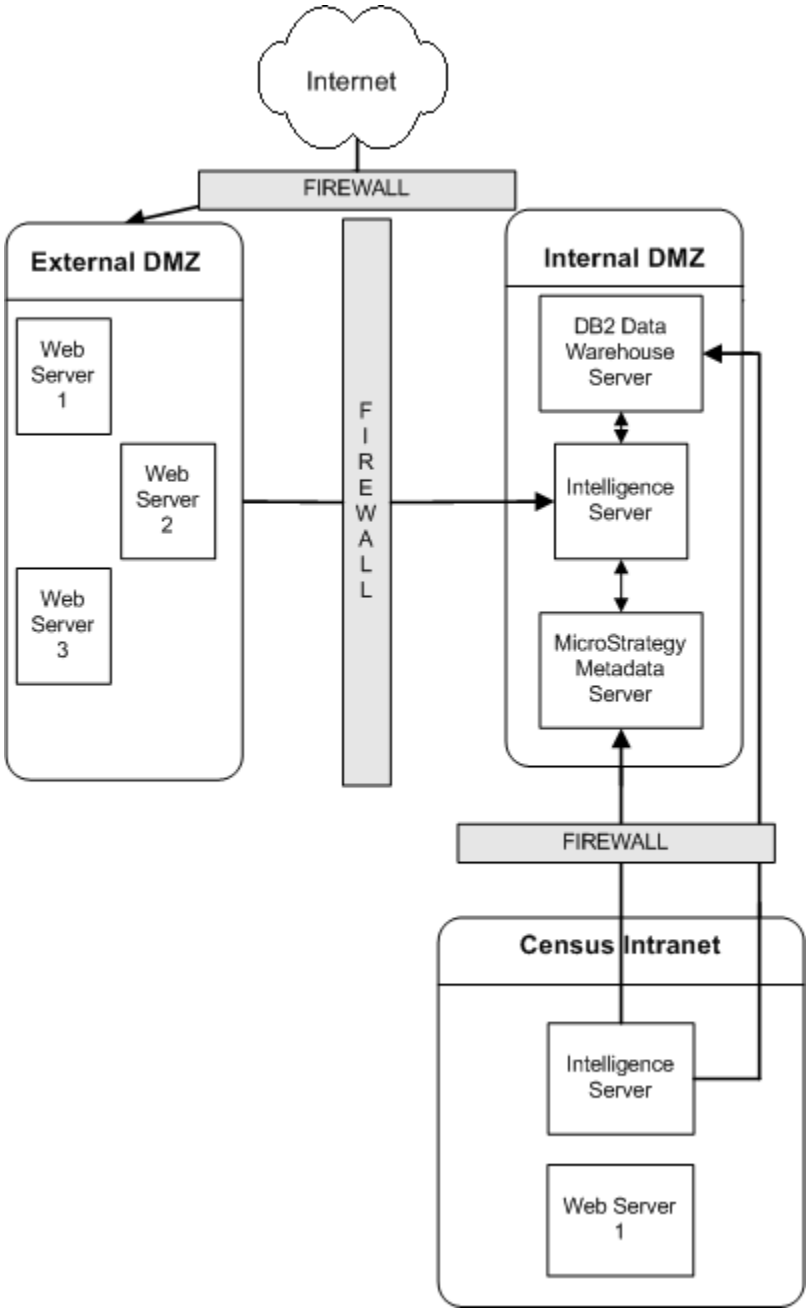
Two separate DMZs were established in the Bowie hosting center to secure AQ system components for the Internet site behind a firewall. The **External DMZ**, was configured to house web servers allowing public Internet access to AQ. The **Internal DMZ**, was configured to protect the Intelligence servers and database servers for the public AQ site behind a firewall. A port was opened on the firewall to permit communication to and from the web servers and Intelligence servers. No direct access to the data warehouse and MircoStrategy metadata servers is possible since these communication ports were not opened on the firewall. In addition, tampering with the way filters are applied to data is prevented since the Intelligence server is behind a firewall.

The Intelligence server uses ODBC protocol to connect to the DB2 UDB data warehouse server and MicroStrategy Metadata database. The ODBC accounts and user accounts are stored in the MicroStrategy Metadata database and the passwords are encrypted with 128-bit TEA algorithm. **Table 10** summarizes the different types of user id and password security mechanisms employed in the MicroStrategy environment:

Table 10: Account Security

User	Purpose	Password Security
MicroStrategy Metadata User	DB2 user id for MicroStrategy Intelligence Server to connect to the MicroStrategy Metadata database instance.	Password is encrypted with 128-bit TEA algorithm and stored in the Windows NT registry.
MicroStrategy Database User	DB2 user id for MicroStrategy application users to connect to Data Warehouse database instance. Each application user can have their own user id or share a generic one.	Password is encrypted with 128-bit TEA algorithm and stored in the MicroStrategy Metadata database.
MicroStrategy Application User	MicroStrategy user id for application users to access web applications that connect to the Intelligence server.	Password is encrypted with 128-bit TEA algorithm and stored in the MicroStrategy Metadata database.
MicroStrategy Administrator User	MicroStrategy user id with administrative rights on the Intelligence server and Metadata.	Password is encrypted with 128-bit TEA algorithm and stored in the MicroStrategy Metadata database.

Figure 26: AQ Network Architecture



For a detailed Network Architecture Diagram please refer to:

- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the High Level System Architecture.*
- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the DADS DMZ Visio Diagram.*

5.2. Authentication

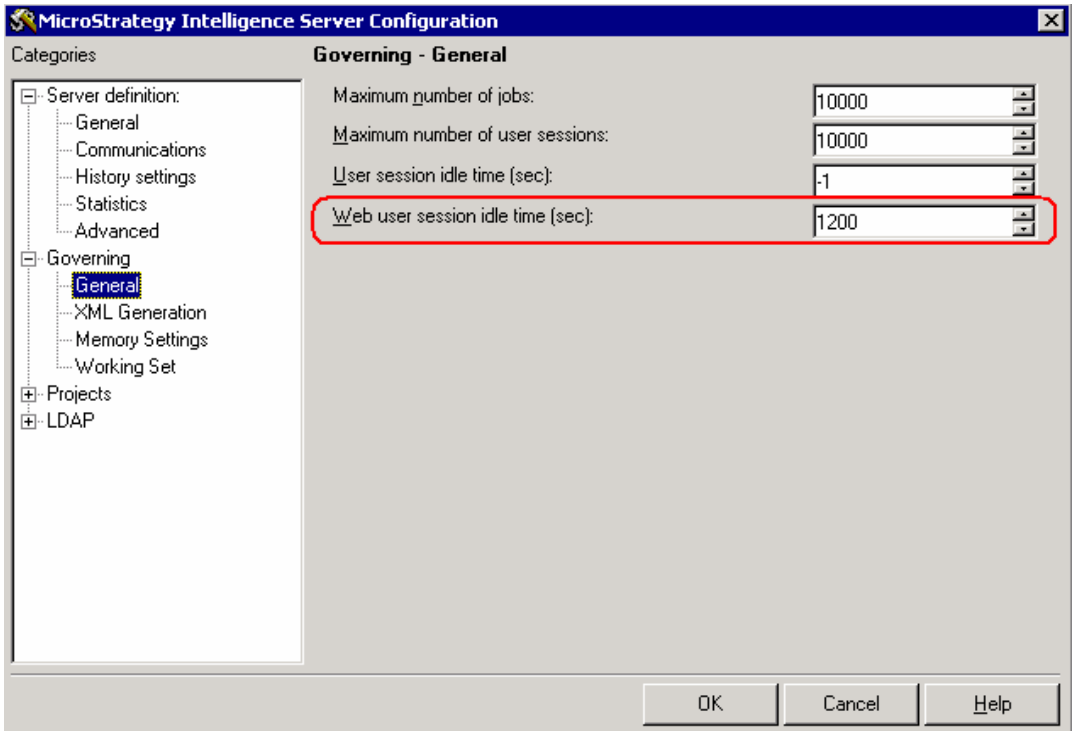
All AQ environments require a login whether they are production, PA or development. User accounts are created on the Intelligence server and stored in the MicroStrategy metadata database for that Intelligence server. By configuring a web server to use a specific Intelligence server, the server administrator causes the web site to inherit the accounts created on that Intelligence server. Therefore, it is important to keep the Intelligence server used for the public facing web site separate from the unfiltered intranet site. The user's userID and password are written to the client in an encrypted temporary cookie. Only the Administrator can turn off this option modifying preferences from the AQ web site. HTTPS is not currently used to encrypt logins from the external DMZ to the internal DMZ, but could be enabled on the web servers if desired.

For details on creating user account see section 7 User Management.

6. SESSION MANAGEMENT

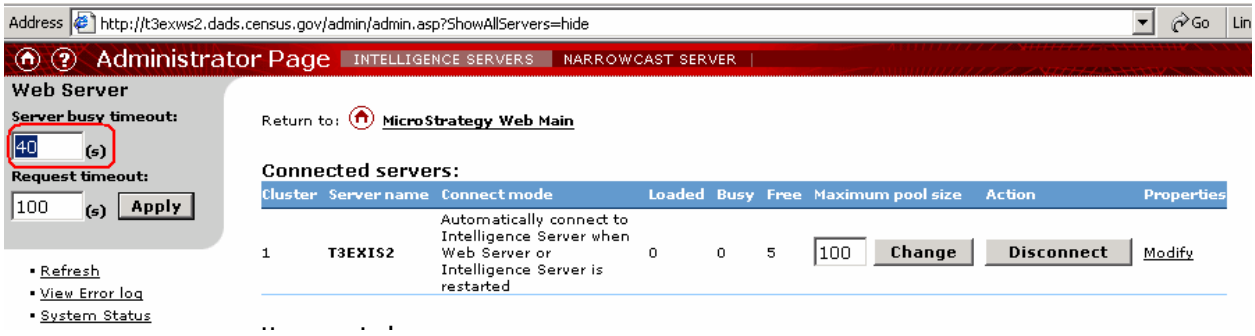
User sessions are primarily managed by the MicroStrategy Intelligence server. The Intelligence server issues a session id to a user upon successful login and includes this session id with every user request in order to maintain state in the application. Sessions are configured to expire after 1200 seconds (20 minutes) of inactivity on the AQ site by setting the "Web User idle session idle time" (see **Figure 27**).

Figure 27: Web Server Timeout Value on Intelligence Server



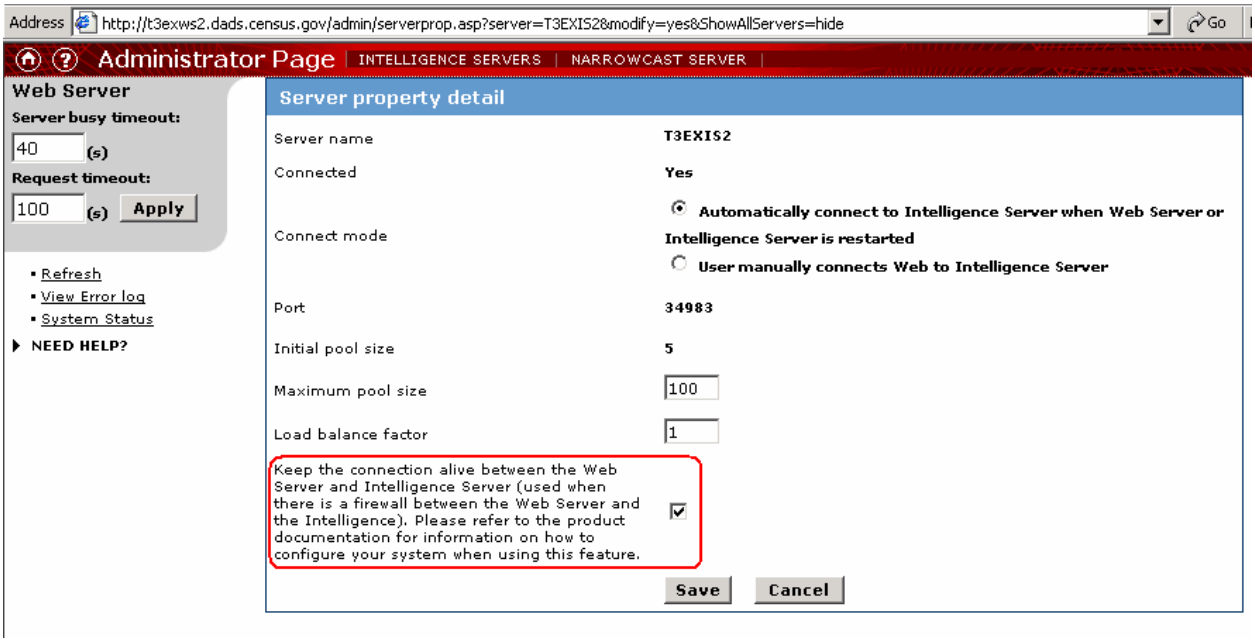
However, Microsoft's Active Server Page architecture has its own session timeout management and can independently terminate sessions after a period of inactivity on an ASP page. To counteract this, MicroStrategy recommends either modifying the ASP timeout value in the web server's registry or setting an additional timeout using MicroStrategy's Web Administration Page. The AQ system sets a web server timeout value of 40 seconds using MicroStrategy's Web Server Administration Page (see **Figure 28**).

Figure 28: Web Server Administration Page Timeout Value



The Census Bureau's firewall is set to time out after a period of inactivity. In order to counteract this, the keep connection alive option is checked on the MicroStrategy Web Server Administration page. When this setting is enabled, the MicroStrategy Web server will send a signal to the Intelligence server on each open TCP connection periodically, so that the firewall will not see the connection as idle. The setting is located in the MicroStrategy Web "Server Properties" administrator page, as illustrated in **Figure 29** below.

Figure 29: Keep Connection Alive



7. USER MANAGEMENT

Users logins and passwords are administered from MicroStrategy's User Manager on the Intelligence server and are stored in the DB2 metadata database. A batch of user accounts can be added by using MicroStrategy's Command Manager. New users should be added to the "Everyone" group with a Security Role Selection of "+ Inherited Access." This security role is configured to prevent ordinary users from running administrative tasks within the AQ web site such as viewing the SQL that generated cross-tabulation. Administrators are also created within the Everyone Group, but their Security Role Selection for all projects is "Power Users + Inherited Access."

Power Users like the Administrator can view the SQL that was run to cross-tabulate data by selecting "View Tabulation Results" from the results page, ordinary users can't view SQL. Users added via User Manager inherit access to all projects on that Intelligence server, unless project access rights are modified from the Project Access tab in User Manager. By convention, user logins are either a Census Bureau "James Bond ID" or a login specified by the external user.

Figure 30 below depicts some of the Project Access Rights configured for a user.

Figure 30: MicroStrategy User Management Settings

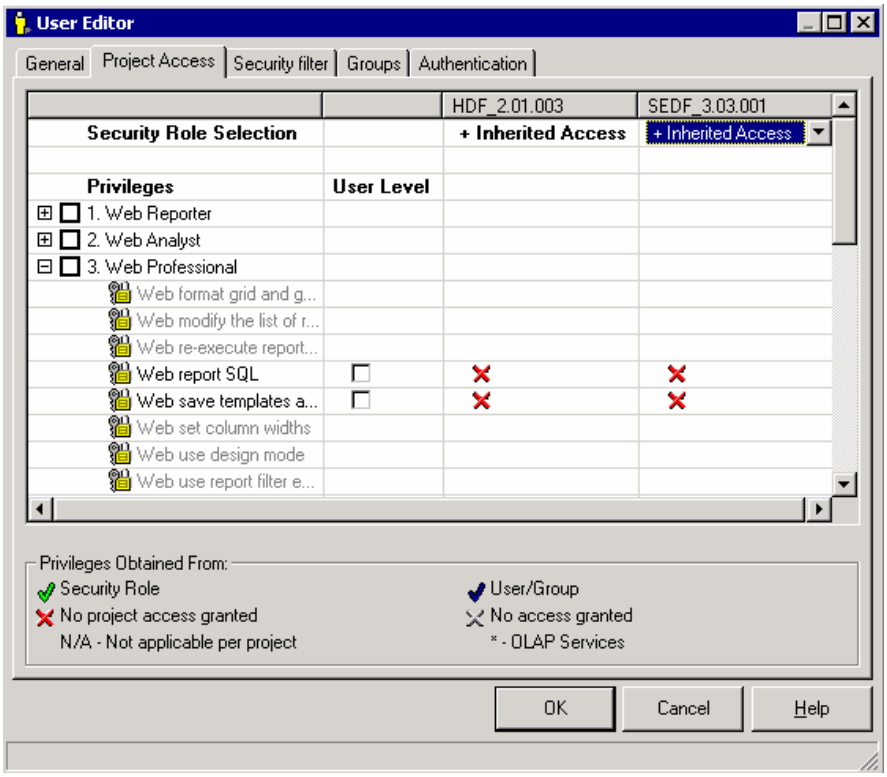
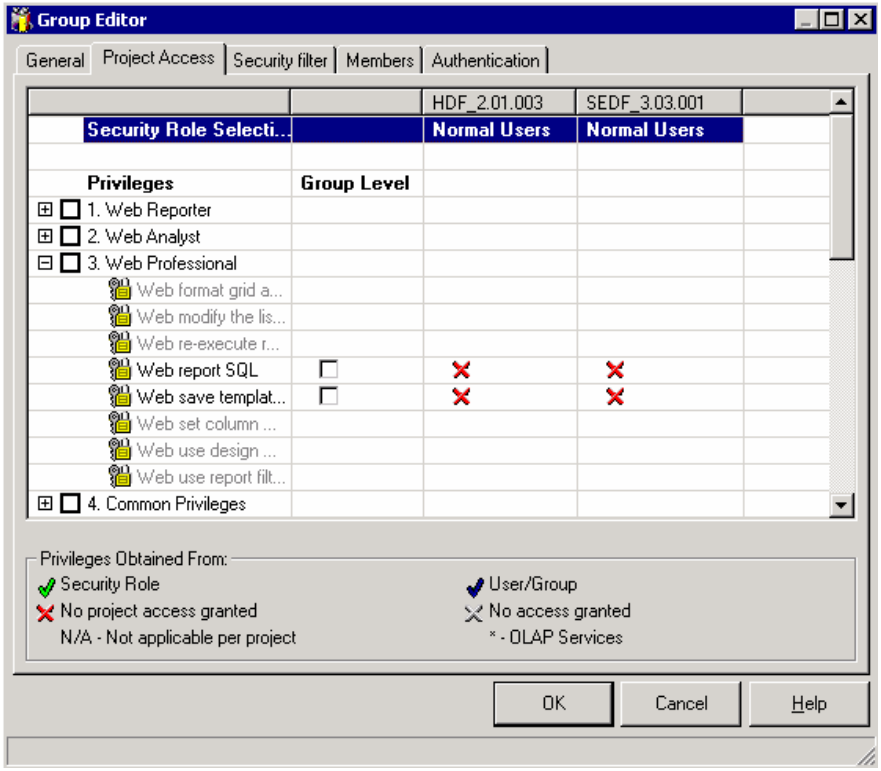


Figure 31 below depicts some of the Project Access Rights configured for the “Everyone” group.

Figure 31: Project Access Rights for "Everyone" Group



The out-of-the box implementation of MicroStrategy prohibits use of brackets, double quotes or backslashes in a userID and limits the length of the userID to between 1 and 250 characters and is not case sensitive. Passwords can be blank and have no maximum length. These criteria have not been customized in AQ.

For additional details on this topic:

- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the MicroStrategy Administrator, Intelligence Server, and Web Administrator Guide.*

7.1. Single Sign-On

The AQ application was configured to enable single sign-on to both the SEDF and HDF project. When a user logs in, he/she gains access to both SEDF and HDF. Single Sign-On was implemented in AQ by logging users into the SEDF project and then transparently forwarding the login criteria to HDF. Although this has a positive impact on the user experience, it also affects usage reporting because every HDF login is accompanied by a corresponding SEDF login even if the SEDF project was not used during a user session. The result is that login statistics for SEDF may appear slightly inflated compared to what the user actually intended to do.

8. PERFORMANCE DESIGN

Due to the massive amount of HDF and SEDF data that needs to be aggregated in order to produce a result set, performance was a key concern through out the design and implementation of AQ. Several key design decisions were made to improve response time.

First, the design team off-loaded the creation of universe and geographic selection from the Intelligence server to static XML files. By pre-building these menus, users were not delayed waiting for query results to return configuration information. In addition, parsing the XML files was off-loaded from the ASP pages to a custom-coded DLL, Census.dll, which was able to read the XML into a DOM object and store it in memory for the entire application. In this way, the design saved users from waiting for the ASP engine to process XML files for each login.

Second, the AQ design incorporated caching in order to speed response time for commonly requested demographic statistics. By caching previously executed reports, AQ is able to quickly provide the same information to users who request the same geographic and demographic variables and save database resources. In addition, element caching was turned on for geographic attributes to reduce the number of select statements against the data warehouse. All cached objects are configured to expire 24 hours after creation and are stored in the default MicroStrategy cache folder, *./Caches/<server name>,* relative to the Intelligence Server installation directory.

Third, the data warehouse was implemented as in a Star-Snowflake hybrid. Star and Snowflake schemas are highly de-normalized in order to reduce the number of joins required and therefore the amount of time a query takes to execute.

9. SYSTEM MAINTENANCE

9.1. Deployment procedures

The deployment process for the AQ code base involves both the UI code, which is ASP-based and the Intelligence server code, which is object-oriented, but stored in the MicroStrategy Metadata database. The term “build” is inappropriate to use when describing the deployment process for AQ because ASP pages use an interpreted language. In addition, objects on the intelligence server are not compiled during the deployment, but rather they are migrated from one database to another.

The AQ deployment process utilizes MicroStrategy’s duplication tool, which copies the entire metadata database from one environment to another (i.e. pa to production). As part of the duplication process, user accounts are never migrated since accounts are environment specific. Thus, other means of migration, like taking a database dump from one environment and bringing it online in another, are not appropriate.

For additional details on this topic:

- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the Advanced Query Deployment Procedures.*

9.2. System Monitoring and Reporting

9.2.1. System Health

System availability and exception conditions are monitored in AQ using IBM Director 3.1 and Concord eHealth 5.7. IBM Director is primarily used to send critical system alerts to a system administrator and automate remote events like server reboots. Concord eHealth is used to generate more detailed monthly system utilization reports.

IBM Director is configured to monitor only the Windows platform machines (web servers and Intelligence servers) in the AQ environment. All AQ web servers and Intelligence servers are registered as IBM Director Agents with the IBM Director Server, which is installed on a dedicated machine in Suitland, MD. IBM Director is configured to capture hardware and software configuration information from remote machines as well as email and page a system administrator when CPU, Memory, or Hard Disk Space fall below a critical threshold. In addition, IBM Director is used to schedule a weekly reboot of all AQ web and Intelligence servers, which is necessary to ensure optimal memory is available to the application. Over time, the MicroStrategy application may use up free memory without releasing resources. A weekly reboot is therefore necessary to free resources.

Concord eHealth is configured to monitor Windows and AIX platform machines in the AQ environment. All AQ servers are registered as Concord agents with a Corcord Server, which is installed and run by the Census Bureau’s Bowie hosting center. Concord logs statistics in its own format, allowing reporting on those statistics to cover any time frame indicated by the administrator. Concord monitors the following measures of system health:

- CPU Utilization
- CPU Load Average
- System Availability (% of time server responds to a ping)
- Physical Memory (bytes)
- Virtual Memory (bytes)
- Page Faults (page faults/sec)
- Page Scan Rate (pages/sec)
- Disk I/O (reads&writes/sec)
- Disk I/O Busy Utilization
- Disk Errors (errors/sec)

- User Partition Space Used (bytes)
- System Partition Space Used (bytes)
- Total Network I/0 (frames/sec)
- Total Network Errors (errors/sec)
- System Latency (msec)

9.2.2. Auditing Policies

Windows Platform AQ servers in the AQ environment (web servers and Intelligence Servers) are configured to log Windows “Security” events that violate audit policies set up on Windows Machines. The audit policies set up for AQ are designed to ensure that there is an audit trail for logins and use of system resources. Audit policies are configured for the following actions:

- Account logon events
- Account management
- Failure of system events
- Policy change
- Privilege use
- Object access
- Restart, shutdown and system

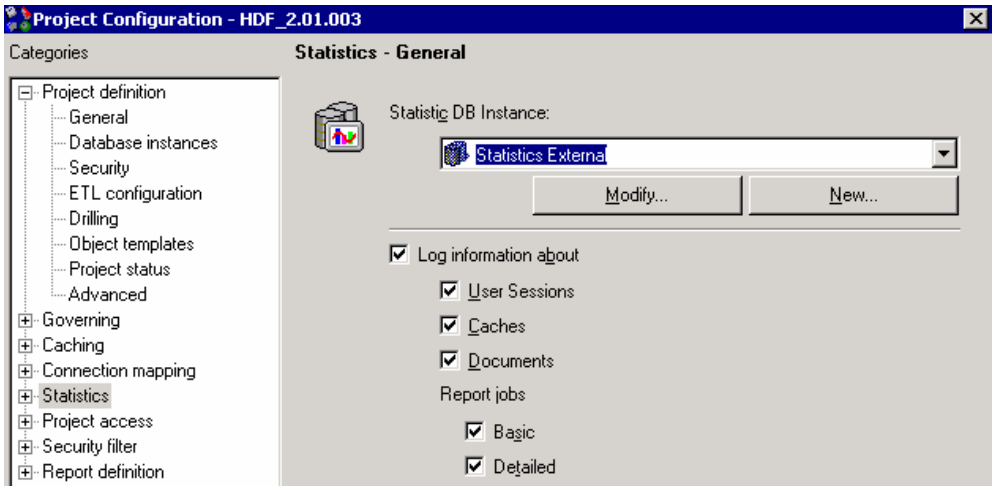
Only Administrators are configured to view the security log, shut down servers from a remote system and run debug programs. As a matter of administration practice, the AQ system administrator logs into the box daily to review the security logs.

- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the MicroStrategy AQ Administration Web site.*

9.2.3. User Statistic Reports

Statistics about AQ system usage are collected via a MicroStrategy-delivered reporting feature that writes usage statistics to the DB2 metadata database. MicroStrategy's statistics reporting feature is configured from the Intelligence Server in the Project Configuration window as shown in **Figure 32** below.

Figure 32: AQ usage statistics configuration



The out of the box MicroStrategy solution collects the following data:

- User Sessions - including logins/logouts
- Report Jobs – including cross-tabulation reports, non-disclosure reports, reports displayed from cache, scheduled jobs, SQL statements run, security filters applied.

No customizations were made to MicroStrategy's user statistics logging feature. However, the statistics are written to the MicroStrategy Metadata database as they are collected and are not in a reportable format, so they must be queried from the database and formatted by a system administrator.

Weekly user statistic reports are generated by querying MicroStrategy metadata database tables. Based on Census Bureau requirements, user statistics reports include information on who has logged in, how many times they have logged in, how many times they ran cross-tabulations and non-disclosure reports. This is a small fraction of all the information collected by MicroStrategy's statistics logging feature. The SQL queries and the information obtained from them, as required by the Census Bureau, are listed in **Table 11**.

Table 11: AQ User Statistic Report Queries

	Report Information	SQL
1	Create temporary tables	declare global temporary table session.logins (Login VARCHAR(128), Logins INTEGER, Date VARCHAR(20)) on commit preserve rows not logged; declare global temporary table session.tabulations (Login VARCHAR(128), Tabulations INTEGER, Date VARCHAR(20)) on commit preserve rows not logged; declare global temporary table session.nonqualify (Login VARCHAR(128), NonQualifying INTEGER, Date VARCHAR(20)) on commit preserve rows not logged;
2	List the users who have logged into a project during a particular time frame. List the number of sessions they opened, how many queries they viewed and how many non-disclosure reports they viewed	INSERT INTO session.logins select b.login, count(*), date(a.eventtime) from IS_SESSION_STATS a, DSSMDUSRACCT b where a.userid = b.object_id and date(a.eventtime) > date('07-30-2004') and date(a.eventtime) < CURRENT DATE and EventNotes like '%Login%' group by date(a.eventtime), b.login order by date(a.eventtime), b.login ; INSERT INTO session.tabulations select a.login, count(b.reportid), date(b.recordtime) from DSSMDUSRACCT a, IS_REPORT_STATS b where b.promptindicator in (6) and date(b.recordtime) > date('07-30-2004') and date(b.recordtime) < CURRENT DATE and a.object_id = b.userid group by date(b.recordtime), a.login order by date(b.recordtime), a.login; INSERT INTO session.nonqualify select a.login, count(b.reportid), date(b.recordtime) from DSSMDUSRACCT a, IS_REPORT_STATS b where b.promptindicator in (4) and date(b.recordtime) > date('07-30-2004')and date(b.recordtime) < CURRENT DATE and a.object_id = b.userid group by date(b.recordtime), a.login order by date(b.recordtime), a.login; select L.date as Date, SUBSTR(L.Login,1,15) as User, L.logins as Sessions, T.tabulations as tabulations, N.nonqualifying as Nonqualify from session.logins L LEFT OUTER JOIN session.tabulations T ON L.login=T.login and L.date=T.date LEFT OUTER JOIN session.nonqualify N ON L.login = N.login and L.date = N.date order by date, User;
3	Drop temporary tables	Drop table session.logins; Drop table session.tabulations; Drop table session.nonqualify;

In addition to weekly reports summarizing how many times a user logged in and how many queries and non-disclosure reports the user ran, the Census Bureau occasionally requires reports detailing which cross-tabulations users ran and what the non-disclosure results were. For example, the Census Bureau may want to know that 90% of users ran a query to find information on people with disabilities in Alameda County, California and could not view results due to disclosure rules. Such information is used by the Census Bureau to assess whether or not the AQ application should be modified to allow more data to pass disclosure. This so-called “SRD Report,” is not generated from the statistics logged by the out of the box MicroStrategy solution. Instead, the ASP page code is customized to write each user request to individual XML log files on the Web Server. A stand-alone VB application called “CensusSRD,” was written to process these XML files and produce an MS Excel spreadsheet listing each request and its disclosure results.

Additional details on MicroStrategy’s User Statistics can be found in:

- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the MicroStrategy Administer, Intelligence Server, and Web Server Guide.*

- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the MicroStrategy Installation and Configuration Guide.*

The source code and installation files for the Census SRD VB application are located in the "TIER3" project in ClearCase under the "/UI/CustomApplications" folder. Please refer to the Advanced Query Deployment Procedures:

- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the Advanced Query Deployment Procedures.*

Instructions for running the Census SRD VB application can be found in:

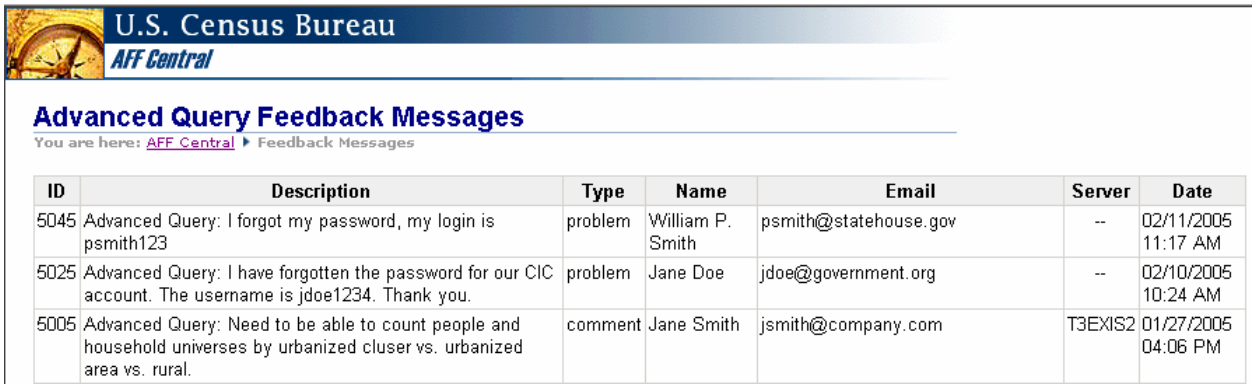
- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the About Census SRD document.*

9.2.4. User Feedback Reports

In order to continually improve the Advanced Query system, user feedback is collected from a form on the AQ web site. Feedback is written to and stored in an external Oracle database, not in the DB2 data warehouse or DB2 metadata database. The decision to store AQ feedback on an Oracle database server was motivated by the desire to consolidate feedback from all online systems and thereby make reporting easier. The American FactFinder system was already using an Oracle database instance to store feedback, so an instance was created on the same server for AQ feedback.

A PERL script, aq_fdbk.pl, pulls feedback from the Oracle database and displays it on a web page (see **Figure 33**). AQ feedback can be viewed from a URL on the BOC intranet, http://affreview1.dads.census.gov/fdbk/aq_fdbk.pl.

Figure 33: Sample AQ Feedback Report



U.S. Census Bureau
AFF Central

Advanced Query Feedback Messages

You are here: AFF Central ▸ Feedback Messages

ID	Description	Type	Name	Email	Server	Date
5045	Advanced Query: I forgot my password, my login is psmith123	problem	William P. Smith	psmith@statehouse.gov	--	02/11/2005 11:17 AM
5025	Advanced Query: I have forgotten the password for our CIC account. The username is jdoe1234. Thank you.	problem	Jane Doe	jdoe@government.org	--	02/10/2005 10:24 AM
5005	Advanced Query: Need to be able to count people and household universes by urbanized cluser vs. urbanized area vs. rural.	comment	Jane Smith	jsmith@company.com	T3EXIS2	01/27/2005 04:06 PM

9.2.5. Report Execution Time

Report execution time is monitored in AQ to provide a mechanism for continuously fine-tuning the database and improving performance. A Korn Shell script, aqMonitor.ksh, is installed on tier3db1 to send an email alert to system administrators if any report takes longer than 20 minutes to complete. Since every cross-tabulation is associated with multiple queries, report execution time is the sum of the time each of these queries took. The shell script connects to the MicroStrategy metadata database to evaluate report jobs and extract the following information:

- Job ID
- Execution Time (in Minutes)
- SQL Statements
- Login and Name of user who ran the report.

The aqMonitor script is configured to run every morning at 1:00 am and report on queries run the previous day in AQ. Email alerts are titled "Warning! AQ Monitor found 1 long running query."

The source code and installation information for aqMonitor.ksh can checked out from the “TIER3” project in ClearCase under the “META\Admin Scripts” folder. For instructions on joining the “TIER3” project, please refer to the Advanced Query Deployment Procedures:

- *See section **Error! Reference source not found. Error! Reference source not found.** for the pointer to the Advanced Query Deployment Procedures.*

10. KNOWN ISSUES

IBM regards risk management as an important consideration in application design. The Advanced Query design was crafted to limit application downtime and single points of failure within the limits of hardware resources provided.

10.1. System Monitoring

The AQ system is monitored to ensure the health of individual system components, however transaction monitoring to ensure end-to-end functionality and compliance with service level agreements is not implemented.

10.2. Single Points of Failure

There are several single points of failure in the Advanced Query production environment. Neither the MicroStrategy metadata database server nor the database server has a fail-over source should they go down. In addition, although there are multiple Intelligence servers in the production environment, they are not load-balanced or configured to fail-over; so the application will not automatically run if the live Intelligence server goes down.

10.3. Deployment Time

Currently it typically takes over 10 hours to migrate the SEDF project Intelligence server code from one environment to another using MicroStrategy's duplication tool. The duplication tool not only copies data from one metadata database to another, but also does extensive schema and validity checks. This process is lengthy in the SEDF project because there are a vast number of variables and tables from which those variables are derived. Any interruption in the network connectivity poses the risk of corrupting a migration.

10.4. Intelligence Server Clustering

Currently, the three production Intelligence Servers are not clustered. Although MicroStrategy 7.2.2 supports clustering between Intelligence Servers to increase the load a production system can handle, the development team found that when attempting to cluster more than two servers, the user request was not always forwarded to the same Intelligence Server and the user's state was lost. As a result the team removed clustering and decided to operate the production environment using only one active Intelligence Server. Currently this is sufficient because there are approximately 700 production users. However, if the application were to be deployed to Federal Depository Libraries, the user base would jump to 15,000 and it would be necessary to reconsider clustering. One solution could be to cluster production Intelligence Servers in pairs.

11. APPENDIX

11.1. Acronyms

Acronym	Explanation
API	Application Programming Interface. Enables one program to use facilities provided by another
ASP	Microsoft Active Server Pages
AQ	Advanced Query Application
BOC	Bureau Of Census
COM environment	A Microsoft Windows framework for sharing modular objects.
DLL	Dynamic Link Library
DMZ	Demilitarized Zone
DOM object	Document Object Model, a set of standards for representing and XML file as a 'DOM' object.
HDF	Hundred Percent Detail File (Census 2000)
HTTP	Hyper Text Transfer Protocol; the WWW protocol that performs the request and retrieve functions of a server.
HTTPS	HTTP Over SSL. Protocol enabling the secured transmission of Web pages
Jam value	A hard coded value to substitute in place of a calculated value
JSP	Java Server Page
ODBC	Open Database Connectivity. Standardized interface for accessing a database from a program.
PERL	Practical Extraction and Report Language. An interpreted programming language.
SEDF	Sample Edited Data File (Census 2000)
TEA	Tiny Encryption Algorithm is a cryptographic algorithm that uses a 128-bit crypto key to encrypt sensitive data.
XML	Extensible Markup Language

Table 12: Acronyms and their explanation.

12. INDEX

12.1. Index of Tables

12.2. Index of Figures